UCA007-727.M

GOVP 12010440

100

조직배양실 소식물체 핸들링 시스템 개발

Development of Plantlet Handling System for Tissue Cultivation Room

연구기관

한국기계연구원

농 림 부


대한민국
기증등록
2000. 8. 03
830060
국회도서관

제 출 문

농림부 장관 귀하

본 보고서를 "조직배양실 소식물체 핸들링 시스템 개발" 과제의 최종 보고서를 제출합니다.

1999. 10.

주관연구기관명 : 한국기계연구원
총괄연구책임자 : 박 경 택
연 구 원 : 최 성 락

요 약 문

I. 제 목

조직배양실 소식물체 핸들링 시스템 개발

II. 연구개발의 목적 및 중요성

조직 배양실에서 대량생산을 목적으로 하는 소식물체 증식작업 공정을 자동화하는데 필요한 소식물체 핸들링 시스템 개발을 목표로 한다. 현재 이 작업은 수작업으로 이루어지고 있으며, 작업자 및 주변환경에 의한 오염에 때문에 증식작업의 증식재배 불량율이 10 ~ 20 % 정도로 아주 높다. 이 작업 공정을 자동화함으로 인력절감을 할 수 있고, 작업자에 의한 오염을 배제함으로서 청정 작업을 할 수 있으므로 증식재배 불량율을 낮출 수가 있다. 이렇게 함으로서 전체적으로 증식작업의 생산성을 향상시킬 수가 있다.

III. 연구개발의 내용 및 범위

- 소식물체 인식 시스템
  - 씨감자 배양방법 및 배양용기 형상
  - 증식작업 자동화를 위한 화상처리 및 조명장치
  - 핸들링 자동화를 위한 소식물체 인식 알고리즘
  - 소식물체 절단 및 피킹 위치 결정 알고리즘

■ 소식물체 핸들링 시스템

- 핸들링 자동화를 위한 절단 및 그리핑 메카니즘
- 소식물체 핸들링 로봇 시스템
- 핸들링 시스템 통합 및 성능 실험
- 개발시스템에 대한 경제성 검토

IV. 연구개발의 결과 및 활용에 대한 건의

- 조직배양실 작업의 자동화를 위해 작업환경의 표준화가 필요
- 자동화에 적합하고 생산성을 갖는 씨감자 배양방법 및 배양용기
- 조직배양실 소식물체 증식작업용 자동화 핸들링 시스템
- 핸들링 시스템의 상품화를 위한 관련주변 시스템 개발 및 작업 공정의 청정화에 대한 추가 연구 필요
- 경제적으로 타당성을 갖는 생산규모 및 자동화 시스템 필요

SUMMARY

I. Title

Development of Plantlet Handling System for Tissue Cultivation Room

II. Objective and Importance of Research and Development

The objective of this research is to develop the automatic system to handle plantlet in tissue cultivation room in order to make automation of propagating process in mass production. Currently, this process is done by manual process, and so failure of propagation is about 10~20 % due to contamination of worker and working environments. The Automation of propagation process can exclude worker and clean the working room, and so the failure of propagation can be decreased. By automation of propagating process in mass production, the productivity of propagation can be improved.

III. Contents and Scope of Research and Development

- Recognition system of plantlet in tissue cultivation room
  - Propagating method of seed potato and shape of petri dish
  - Study on image processing and lighting source
  - Recognition algorithm of plantlet
  - Algorithm of determination of cutting and griping position

■ Plantlet handling system

- Mechanism of cutting and griping
- Handling robot system
- Integration and test of handling system
- Economical evaluation of commercialization of developed system

IV. Result and Application of Research and Development

- Standing erect method is not proper to propagation of seed potato in tissue cultivation room
- Crawl method is proper to automation of propagation of seed potato in tissue cultivation room
- Plantlet recognition and determination of cutting position by image processing for handling robot
- To commercialize the developed system, additional development is needed about cleanization of the related system and propagating process
- In the case of mass production of propagation, automation is to have economical propriety

INDEX

목 차

제 1 장 서 론

제 1 절 연구개발의 목적 및 범위

식물 조직배양에 의한 묘 생산은 화훼류, 채소, 과수 분야 등에서 많이 이루어지고 있으며, 농업과 원예 분야에서 중요한 위치를 차지하고 있다. 식물 조직배양에 의한 번식, Micropropagation 기술에 의한 묘 생산은 다른 방법들 보다 우수한 묘 배양을 생산 할 수 있다. 그러나, Micropropagation에 의한 배양 묘 생산의 비용은 실생 묘 생산과 삽수 묘 생산 방법보다 비용이 많이 들기 때문에 실제 이용이 제한이 되고 있다. Micropropagation의 높은 생산비용은 증식작업에 소요되는 높은 인건비, 증식과정에서의 낮은 생산속도, 순화 과정에서의 낮은 생존율에 기인하고 있다. 높은 인건비, 낮은 생존율에 해당하는 문제를 해결하기 위해 증식 과정에서의 수작업 과정을 기계화 및 자동화함으로서 해결할 수가 있다.

본 연구에서는 생명공학연구소의 인공 씨감자 연구실의 협조 하에서 인공씨감자의 대량생산을 목표로 하는 무인배양 시스템에서의 작업을 기계화 및 자동화하는 것을 1차 목표로 하고, 여기서 축적된 기술을 바탕으로 하여 화훼류 및 일반 특용작물의 증식공정의 자동화를 2차 목표로 하고 있다.

단순 반복작업을 기피하는 현상에 기인하는 인력난과 고임금에 의한 높은 생산단가를 낮추기 위해 수작업 과정을 자동화 시스템으로 대체하여 생산과정을 무인화 할 필요가 있다. 조직 배양실 순화 과정 중 높은 생존율을 위해서 무균 환경이 필수적으로 요구되기 때문에 외부 작업자와 격

리된 완전 무인화 자동생산 체제를 갖춘 청정 생산환경이 필요하다. 국내 농업환경에 적합한 고유의 고부가가치 생산기술에 대한 체계적인 개발이 필요하다. WTO 체제에 따른 농수산물 수입개방으로 산업구조의 재편성이 필요하며 이에 따른 농업 노동력의 양적, 질적 감소 추세에 의해 국내 농업이 위기에 봉착하고 있다. 이를 탈출하기 위해 노동 집약적 농업에서 고부가가치의 기술 집약적 농업으로 방향 전환이 필요하다. WTO 체제 출범으로 인한 농수산물 수입개방에 대한 통상압력이 가중되고 있기 때문에 농업분야의 국제 경쟁력 강화가 필요하다. 국내 전체 인구의 13 %가 농업에 종사하고 있지만 농수산물 수입국으로 되어 있고, 이스라엘의 경우 6 %가 종사하고 있지만 수출국으로 분류되고 있다. 농업노동 생산성을 향상시킬 필요가 절실하고, 3D 업종 기피 현상과 농업 노동력의 여성화 및 고령화에 대한 대책이 필요하다.

선진국에서 첨단 조직 배양 시스템에 관련된 자동화 기술 개발은 주로 고급 작물이나 화훼류 중심으로 이루어지고 있다. 지역적으로 주변에 유통시장이 조성되어 있고 국제공항, 항구, 고속도로 등의 교통 환경이 좋고, 비교적 온화한 기후로 농업의 여건이 좋으면서도 경지 면적이 좁은 유럽 국가들 사이에서 개발을 시작하여 왔다. 이들 유럽 국가들은 처음부터 상업적인 목적으로 기술을 체계적으로 개발하여 왔기 때문에 학술적이기보다는 다분히 실용적인 기술개발 형태로 기업화되어 있다. 세계적으로 대표적인 기술 보유처 및 기술내용은 다음과 같다.

- Phyto Nova (Netherlands) : 조직 배양공정 자동화
- V.C.I. (Netherlands) : 조직 배양용 배양액 조제 자동화
- Hawe (Netherlands) : 온실용 토마토 재배 자동화
- Agri System (Netherlands) : 토마토 생산용 식물 공장 시스템
- Dummen (Germany) : 접목 기술 자동화
- Evtra Ltd (Twain) : 조직 배양용 배양액 조제 자동화
- Fides B.V.(Netherlands) : 이식용 절단기, 접목 자동화 기술
- A.B.O. (Israel) : 온실, 이송, 파종, 수확 자동화

그러나, 이러한 기술들은 유럽국가 자체에서도 보급이 그다지 보편화되어 있지 않아 보급 상태로는 초기 단계를 면하지 못하고 있다. 자동 배양 시스템에 관련된 연구개발에 대한 주요 기초 연구개발들은 다음과 같다.

- Automative Feeding Transfer (Suggs C.W., 미국)
- Air-pruned Transplant Production System for Fully Automated Transplanting (Huang, B.K. 미국, Ai.F. 일본)
- Optimization of Containerized Transplant Production : The Hybrid-Float System (Suggs, C.W. 미국)
- Development of Automatic Transplant using Chain Pot for Vegetable Crops (Namba, T., Tanimura, M. 일본)

현재 국내 대학 및 연구소에서는 실험실 단위의 조직 배양실이 다수 운영되고 있으나 작업의 대부분이 수작업에 의존하고 있다. 예를 들어, 국내 대학 및 연구소에서 개발된 인공 씨감자 대량 생산 방법은 특수한 형태로 제작된 배양용기에서 1단계로 인공 씨감자 대량 생산용 감자 줄기를 번식시킨 후 생육조건을 적절히 조절하여 8 주간의 2단계 배양을 통해 씨감자를 대량으로 생산한다. 이렇게 생산된 인공씨감자는 매우 적은 비용으로 장기 저온저장이 가능하고, 중괴경(Minitube) 또는 육묘 형태로 농가에 저가로 대량 보급이 가능하다. 이 방법은 타 방법에 비해 매우 높은 생산성을 지니고 있으나 실험실 수준 이상의 적용 시도가 이루어지지 않고 있어 산업화를 위하여 기계화 및 첨단 자동화 기술의 도입이 필요하다.

씨감자 이외에 화훼, 임목, 채소, 인삼, 연초, 배, 백합, 딸기, 구기자 및 벼와 같은 식물체를 대상으로 한 유전자 형질 전환, 이의 재분화, 3차 대사 산물 생산, 조직 배양의 실험적 연구 역시 활발하며, 전문 연구팀으로는 고려대, 서울대, 배재대, 충남대, 경북대, 경상대, 충북대, 순천대, 연구기관으로 생명공학 연구소, 인삼연초 연구원, 기업에서는 두산 연구소, 진로

등을 들 수 있다. 이러한 전문팀의 활성화된 연구 노력에 반하여 이의 기계화, 자동화는 아직 기초단계에 머물어 있는 수준이다. 식물 공장 분야에서는 정부의 정책지원에 힘입어 최근 2~3년 사이에 급속한 기술개발 및 연구의 빈도가 높아져 가고 있으며 공정 육묘장의 경우 지난 92년부터 국내 보급이 시작되면서 현재 전국에 약 30개소가 설치 운영되고 있다. 학술적인 연구도 꾸준히 이루어져 서울대, 경상대, 경북대, 성균관대, 건국대 등 관련 대학의 관련 학과에서 기반기술의 구축을 어느 정도 해놓은 상태이다.

조직배양기술에 자동화기술을 적용한 사례는 아직 없으며 자동화 기술체계가 전혀 구축되어 있지 않다고 보아도 좋을 것이다. 현재까지 일부 개발되어 실용화된 기술은 유리온실, 식물공장에 관련한 요소기술들이며, 자동화 핵심기술은 외국 기술에의 의존도가 높고 표준화가 미진하여 국내 연구개발 체계상에 혼란을 주고 있으며, 시스템 관련 기술이 비교적 빈약하게 다루어지고 있다. 이러한 관점에서 본 연구의 조직 배양실에서 소식물체 핸들링 자동화 기술에 관련된 내용은 다음과 같이 3 가지를 고려한 기술개발이 이루어져야 할 것이다. 즉, 기계화 및 자동화의 관점에서 농업에 관련 핵심 요소기술들을 접목시켜야 할 것이다.

- 조직배양실 구성요소의 표준화 및 모듈화 기술
- 조직배양실 환경 및 특성을 고려한 자동화 기술
- 첨단 공장 자동화 기술을 농업 자동화에 적용하는 기술

첨단산업 자동화 기술은 타 산업 분야에서 다년간의 연구를 통해 확보된 기술이며, 농업에 관련된 자동화 기초기술에 대한 본 연구를 진행하였다.

제 2 절 조직배양실의 소식물체 생산

현재 화훼분야에서는 난, 카네이션, 국화, 장미 등과 같은 많은 원예식물들을 식물 조직배양에 의한 번식, 즉 Micropropagation의 기술을 이용함으로서 종래의 실묘 생산 및 삽수 생산기술과 비교해 보다 우수한 품질의 식물 묘를 배양 묘로서 공급할 수 있게 할 수 있다. 또한 Micropropagation이 가능한 종은 해마다 늘어나고 있고 그 상업적 이용이 세계적으로 보급되고 있다. 따라서 앞으로 식물 조직배양의 묘 생산에의 이용은 화훼류뿐만 아니라 채소 과수 분야에서도 이루어져 식물 조직배양에 의한 묘 생산은 농업과 원예 분야에서 중요한 위치를 차지할 것이다. 그러나, Micropropagation에 의한 배양 묘 생산의 경비는 종래의 실묘 생산 또는 삽수 묘 생산과 비교해서 높기 때문에 농업분야의 주요 작물과 조림용 수목 등에는 광범위하게 이용되지 못하고 있다.

식물 조직배양이란 세포, 조직 또는 기관 등을 식물체로부터 무균으로 분리하여 적당한 조건에서 배양하여 생육시키는 기술을 말하고, 본래는 대상 식물체의 형태형성, 대사생리 등을 제어하고 있는 메카니즘의 규명을 목적으로 한 기초연구를 수행하는 연구 분야에서 많이 이용되어 왔다. 그러나 최근에는 그 이용 가치를 최대한으로 창출하기 위한 기술의 확립을 목적으로 하는 실험분야에서 이용하는 경우가 많다. 따라서 식물의 Miropropagation의 확립을 목적으로 하여 실행되어진 연구가 많다.

Micropropagation의 생산비가 높은 것은 높은 인건비, 증식 과정에서의 낮은 생장속도, 그리고 순화 과정에 있어서의 낮은 생존율이 그 원인이다. 여기서 고려해야 할 것은 이러한 문제점은 배양 소식물체의 생장과 발육에 미치는 환경조건의 영향을 충분히 파악해서 해결해 나아가야 한다는 것이다. 그 이유는 배양체가 갖는 유전적 특성을 인간의 활용 목적에

일치하게 하기 위해 높은 효율로 발현시킬 수 있는 기술을 확립할 필요가 있다. 이를 위해서는 배지조성 등의 생화학적인 배양 환경뿐만 아니라 식물체를 둘러싸고 있는 물리적 환경 즉, 온도, 습도, 광조명 등의 환경과 배지의 물리적 환경이 식물체의 생육에 중대한 영향을 미치기 때문이다.

식물 조직배양 과정의 일반적인 흐름도는 그림 1.1과 같다. 이 그림에서 외식편이라고 하는 것은 식물체의 1편(보통 1변이 1~5 mm 정도의 조직편)이고, Callus라는 것은 기능적, 구조적으로 조직화(분화)되어 있지 않는 세포괴, 소식물체라는 것은 경엽이 구별(분화)되어 있는(보통 초장이 10~50 mm 정도) 작은 식물체이다. 또 정식이라고 하는 것은 묘를 밭이나 온실 내의 재배공간에 심는 일이다. 순화라는 것은 배양 묘가 배양기로부터 나온 후와 정식 후에도 순조롭게 생육할 수 있도록 생육환경을 서서히 변화시키는 일이다. 식물 조직배양 과정에 있어서 계대 배양이 되어진 소식물체는 필요에 의하여 증식, 생장, 순화를 시켜서 정식용 묘로 생산되어진다. 여기서 계대 배양이라고 하는 것은 소식물체를 일정 기간씩 재분활함으로서 소식물체를 배양기 내에서 무균적으로 보존, 유지, 그리고 갱신하는 것이다. 여기서 문제는 소식물체의 증식, 생장의 단계, 배양 묘의 순화 단계에 대한 대량증식, 생장 촉진 등이다.

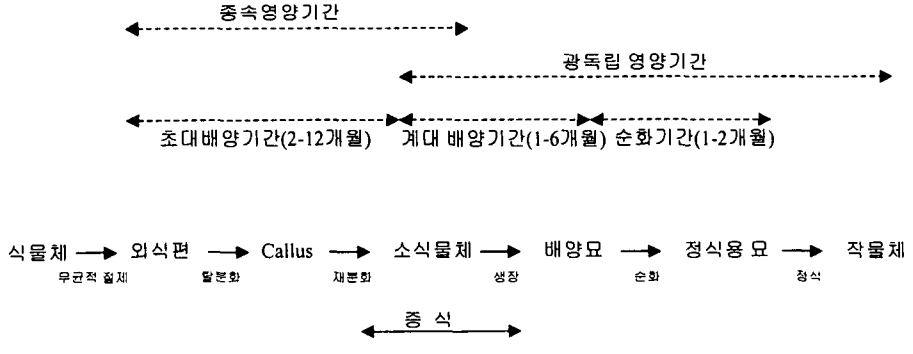


그림 1.1 식물조직배양 과정의 흐름도

일반적으로 소식물체를 배양 묘로 생장시키기 위해서는 1~6 개월을 필요로 하고, 또 배양 묘를 정식용 묘로서 순화시키기 위해서는 1~2 개월이 더 걸린다. 이 순화 기간 중에 많을 경우에는 배양 묘의 20~80%가 고사한다. 고사의 원인으로는 일반적으로 배양 묘는 환경 Stress(특히 수분 Stress)에 약하고, 광합성 능력이 낮기 때문이다. 배양 묘의 생산 Stress의 절감을 위해서는 증식, 생장, 순화 단계의 손 작업을 줄이고, 소식물체를 배양 묘로 하기까지의 기간과 순화기간을 단축시키고, 또 순화 기간 중의 배양 묘의 고사율도 낮추지 않으면 안된다.

현재 실시되고 있는 조직배양을 이용한 묘종 생산은 시험관이나 플라스크를 이용해서 수작업으로 행해지고 있다. 그림 1.2는 조직배양에 의한 묘종의 생산 공정을 나타내지만, 우량인 어미 포기의 생장점을 잘라내어서 재배지위에 컬스(부정세포괴)를 형성시킨 후, 재배지 호르몬 조성을 바꾸어 컬스 위에 다수의 싹을 발아시킨다. 이것을 각각 별도의 재배지에 옮겨 심어 식물체로 분화시키고 분화된 식물체를 또 포기로 나누기도 하고, 잘라내기도 해서 몇 배로 증가시킨다. 이와 같은 작업을 반복해서, 하나의 조직편에서 수천 개, 수만 개의 모종을 생산하고 있다. 이들 작업이 모두 수작업으로, 게다가 크린 벤치 등의 무균 상태에서 실시되기 때문에 묘종의 원가에서 차지하는 인건비의 비율이 70%에 달한다고 한다. 어미 포기에서 채취한 조직편이 생장점이라는 데에도 중요한 의미가 있다. 이것은 바이러스에 조직편을 가지고 오는 것이고, 당연히 그후 작업도 완전히 무균 상태에서 실시된다. 그 결과 생산된 묘종은 어미의 우수한 형질을 갖추고 있을 뿐만 아니라, 바이러스 프리의 우량종이 된다.

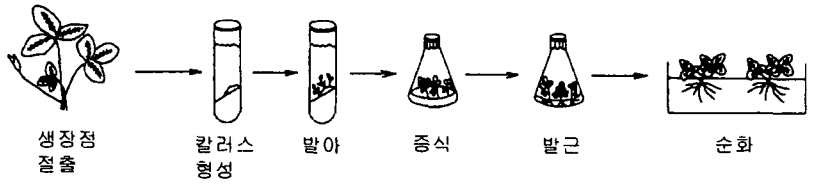


그림 1.2 조직배양에 의한 묘 생산과정

묘종 생산 시스템 개요를 그림 1.3에 나타낸다. 우량종인 친주의 조직(통상은 생장점)을 추출해 내서 조직배양에 따라 식물체로 분화시켜, 자라난 묘종의 마디를 잘라내 다른 재배지로 옮겨 심어 크론을 만들어 이식한다. 꺾꽂이 요령에서 성장과 절단을 반복해서 개체를 수를 늘리고, 바깥 환경에 익숙한 공정(순화)을 거쳐, 농가 등에 묘종을 출하한다.

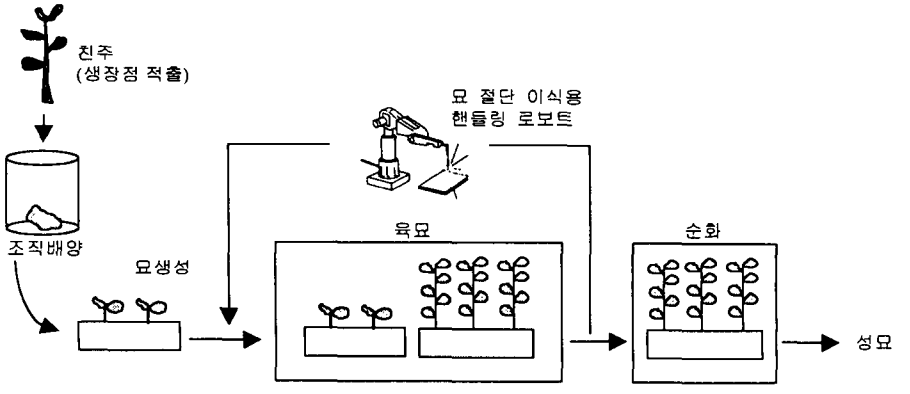


그림 1.3 묘종 생산 시스템

이들의 공정은 식물에 바이러스 혼입이나 곰팡이 등에 의한 오염을 피하기 위해 무균의 환경에서 실시한다. 환경에서는 작업의 거의 모두를

사람의 손에 의존하고 있다. 인간에 의한 오염의 영향이 크기 때문에 무균의 실내에서 묘종을 용기에 넣어 밀폐시키기도 하고, 일부가 오염되어도 전체에 파급하지 못하도록 작은 용기를 다수 준비하고 있다. 자동화하면, 노동력 절감효과는 물론이거니와 그 이상으로 최대의 오염원인 인간을 없애는 장점이 크다. 결국, 묘종 용기를 밀폐할 필요가 없고, 또 큰 용기를 사용해 대량의 모종을 한 번에 배양할 수 있게 되므로, 필연적으로 대량생산으로 향한다. 결과적으로 생산성을 비약적으로 향상시킬 수 있어야 한다.

식물을 재배하는 경우, 일반적으로 종자를 번식 발아시켜 그것을 기르지만, 이 경우 반드시 재배자가 원하는 방향으로 된다고 할 수 없다. 이것은 유전적으로 여러 가지 성질(생물용어로는 형질이라고 한다)을 가진 것이 가능하기 때문이다. 만약 재배자가 원하는 형질을 갖춘 우량종을 싼 값에 입수할 수 있게 된다고 하면, 이것은 농업 발전에 있어서 획기적인 계기가 된다. 현재, 꽃의 일부에 있어서는 이미 이와 같은 묘종으로 재배지가 실용화되어 있다. 우량인 어미 포기 조직편(통상 생장점)을 인공 재배지상에서 배양·증식시키는 조직 배양이라는 기술을 사용해서, 어미포기의 크롬종 묘종을 대량으로 생산해서 양질의 꽃을 만들고 있다. 한편, 바이오 테크놀로지의 발달에 따라 유전자 조작이나 세포융합 등에 의한 식물이 늘어나면 이들 식물은 종자가 열매를 맺는 일이 낮고, 거의 종자를 산출하지 않는다. 따라서, 이들 식물은 조직배양을 이용한 묘종 생산에 의존하지 않을 수 없고, 앞으로 이러한 경우는 계속 증가 할 것이다.

제 3 절 소식물체 증식공정의 자동화

1. 기계화 및 자동화의 필요성

농업 선진국인 유럽, 일본, 미국 등에서 농업의 자동화, 즉 식물 공장 개념을 도입하여 농업의 고부가가치화를 위한 농업의 기계화, 자동화를 위한 연구개발이 지속적으로 많이 추진되고 있다. 특히, 조직 배양실 묘 생산공정을 첨단 자동화 기술을 도입하여 자동화 시스템을 연구개발한 일본의 도시바 생산기술 연구소의 개발 시스템에 대해 알아본다.

유럽에서는 이전부터 묘종만 생산해서 재배농가에 공급하는 기업이 있는데, 현재 국내에서도 묘종 생산의 전문화가 진행 중에 있고, 정부에서도 이것을 적극 추진하고 있다. 이를 위해서는 묘종 생산의 자동화와 대량 생산화가 필요하다. 자동화 기능을 실현해서 실용화에 근접한 식물 공장용 로봇 시스템을 개발하였다. 산업용 로봇과 크게 다른 점은 취급하는 대상물의 위치, 형상을 정확하게 알지 못하기 때문에 대상물의 형상을 인식할 수 있는 기능이 필요한 점과 묘종이 연약해서 연약 정도에 따라 묘종을 살짝 잡아서 절단해 이식하는 절단 및 그리퍼 부분과 시스템 전체 구성에 대해 알아본다.

우량종을 대량으로 생산해서 농가에 공급하려고 하는 움직임이 활발해 지고, 일본 정부에도 묘종 육성 시스템에 대한 연구 프로젝트를 시작시켰다. 이것에 사용되는 주요 요소기구 중 하나가 핸들링 로봇이다. 이것은 조직 배양이라는 바이오 기술로 배양된 묘종의 위치나 형상을 인식해서 적절한 위치에서 절단하여 새로운 재배지에 이식하는 작업을 수행할 수 있는 지능형 핸들링 로봇 시스템이다. 이것은 아직 연구개발 과정에 있고,

소프트 그리퍼(Soft Gripper)와 묘종의 위치검출 방법에 대해서도 알아본다.

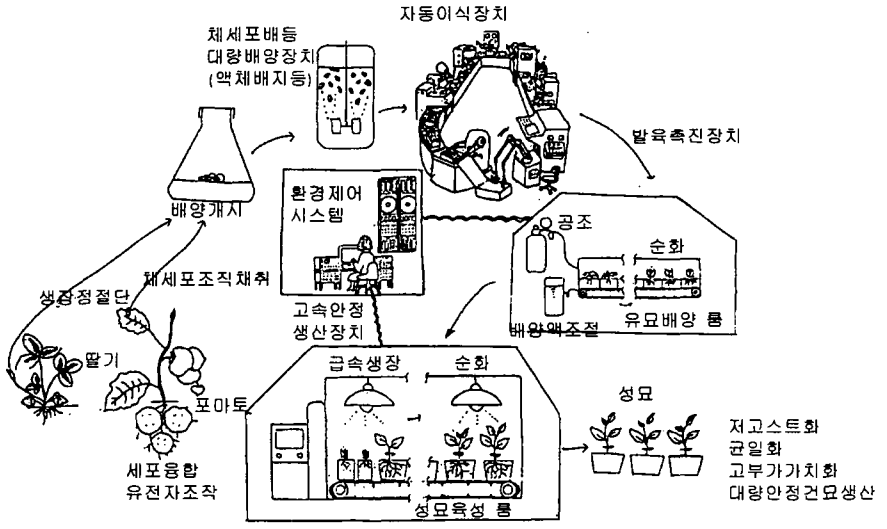


그림 1.4 Bio-Nursery 시스템 개념도

여기서 말하는 묘종 생산이란 종래부터 실시되어 온 밭에서 종자를 심어서 기르는 것과는 전혀 다른 형태이다. 조직 배양이라는 기술을 근거로 산업적으로 그리고 저렴한 비용으로 묘종을 생산할 경우 우선적으로 고려해야 할 것이 인건비의 삭감이다. 현 상태에서 인건비의 대부분은 증식 공정에서 묘종을 절단하여 이식하는 작업에서 발생하고 있기 때문에 이 작업을 로봇화 하는 것이 가장 필요하다.

배양균을 잘라내어 이식하는 공정을 로봇화 하는 것을 노동력 절감 이외에 또 한 가지 이유가 있다. 앞에서 설명했듯이 묘종을 기르는 시스템은 무균상태로 되어야하고, 특히 직접 식물체에 접촉해서 절단하기도 하고 이식하기도 하는 공정을 작업자가 하면, 작업자가 가지고 있는 여러 세균을 묘종에 오염시킬 가능성이 있다. 이러한 점 때문에 무인화, 로봇화가 요구된다.

2. 자동화에 있어서 고려하여야 할 특성

취급하는 대상물이 살아있는 소식물체이기 때문에 종래의 산업용 로봇과는 요구되는 기능이 상당히 상이하다. 또 작업 그 자체도 현재는 작업자가 판단하면서 작업을 수행하고 있기 때문에 이것을 대체할 수 있는 로봇은 상당한 지능 수준을 갖춘 지능형 로봇이 되어야 한다. 이 로봇 시스템이 갖추어야 할 주요 기능은 다음과 같다.

① 로봇이 취급하는 소식물체의 형상이 천연적인 부정형이기 때문에 단순 정형 작업으로는 필요 작업을 수행할 수 없다.

② 소식물체가 있는 위치는 평면적으로도, 입체적으로도 일정하지 않기 때문에 소식물체의 존재 및 위치 감지 기능이 필요하다.

③ 이 로봇의 작업 목적이 소식물체를 적절한 위치에서 절단하여 그 절단 부분의 조직을 이식하는 것에 있기 때문에 소식물체를 절단하는 위치 및 개수 등을 정확하게 판단할 수 있는 기능이 필요하다.

④ 소식물체는 유연하고 약하면서 살아있는 생물체이므로 상처를 입지 않도록 매우 소프트하게 잡을 수 있는 기능이 필요하다.

⑤ 앞에서 설명했듯이 대상이 되는 소식물체가 세균에 의해 오염되는 것을 방지하지 않으면 안되기 때문에 로봇 측의 멸균을 용이하게 할 수 있음과 동시에 만약 세균에 오염된 소식물체를 절단한 경우, 다른 개체로 오염이 확산되는 것을 막는 방법이 필요하다.

⑥ 이상과 같은 기능을 갖추고 또 작업자의 작업량을 상회하는 양의 작업을 할 수 있을 만큼의 고속 작업의 기능을 가지고 있어야 한다.

이상과 같은 기능을 갖춘 로봇 시스템에 대해 연구개발의 목표로 하여 많이 노력하고 있으나 아직도 연구개발을 하는 과정이다.

3. 증식작업의 자동화에 대한 국외 연구사례

1) 소식물체 핸들링 로봇 시스템

일본 도시바 생산기술연구소에서 개발한 묘 증식을 위한 자동화 시스템의 구성은 그림 1.5와 같다. 현재까지 개발중인 주요 부분에 대해 소개한다. 시스템 전체 구성을 살펴보면 크게 두 가지로 나눌 수 있다. 로봇의 동작과 트레이의 공급 및 배출 동작이다. 우선 로봇의 동작이다. 인식 로봇이 묘종의 형상에서 마디 부분을 인식해서 절단 위치를 결정한다. 이 정보를 받아 이식 로봇은 묘종의 파지 → 절단 → 이식 → 파지 → 절단 → 이식 작업 공정을 반복한다. 절단 및 이식 위치는 순차적으로 변해간다.

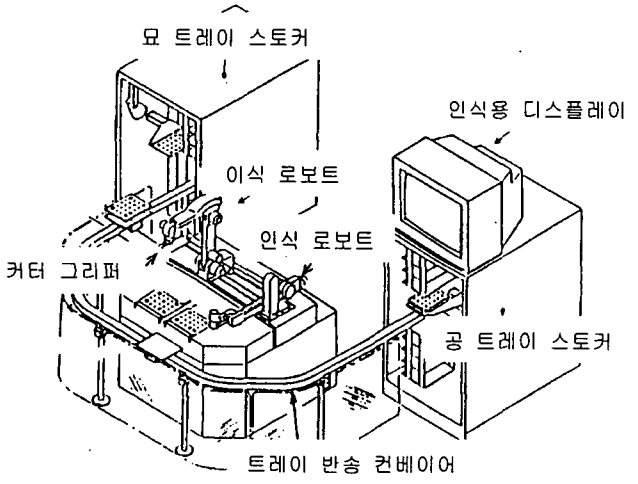


그림 1.5 묘 증식용 로봇 시스템 구성

주요 시스템 구성은 묘의 존재 유무 및 절단 위치 정보를 검출하는 인식 로봇, 묘를 파지 절단하여 새로운 트레이에 이식 작업을 하는 이식 로봇, 작업 트레이를 운반하는 컨베이어, 그리고 작업 트레이를 적재할 수 있는 두 개의 스토커로 구성되어 있다. 이식 로봇의 작업이 완료 되면, 트레이의 공급 및 배출 동작이 시작된다. 트랜스퍼는 작업 완료된 트레이를 컨베이어로 옮겨 실어서 배출 방향의 트레이 적재함으로 이송하여 격납한다. 이것과 병행해서 공급 방향에서 새로운 작업 트레이를 적재함에서 끄집어내서 컨베이어에 실어 로봇 앞까지 운반하는 트랜스퍼로 옮겨 싣는다. 필요한 트레이가 준비되기까지 로봇은 커터 및 그리퍼 부분에 대해 세정 작업을 수행한다. 그림 1.6은 로봇 본체, 소프트 그리퍼, 묘종 검출부 및 커터부에 대한 주요 구성을 보여준다. 전체 시스템 제어는 한 대의 32 Bit 마이크로 컴퓨터를 사용하였고, 계측 데이터는 컨트롤러와 각 기구로의 지령을 보내고 있다.

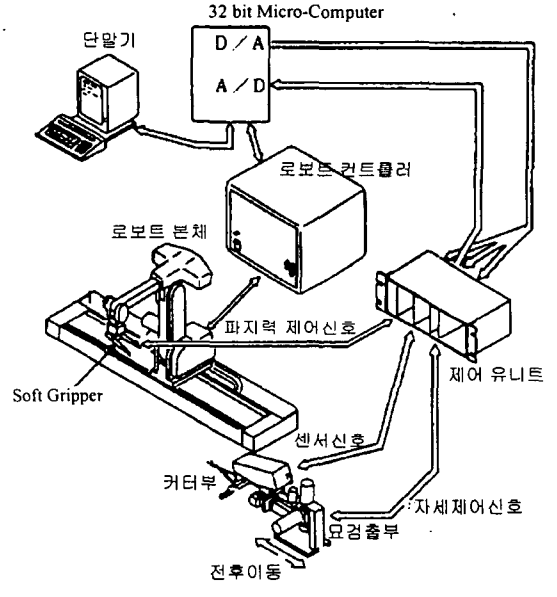


그림 1.6 로봇 시스템 관련 주요 구성도

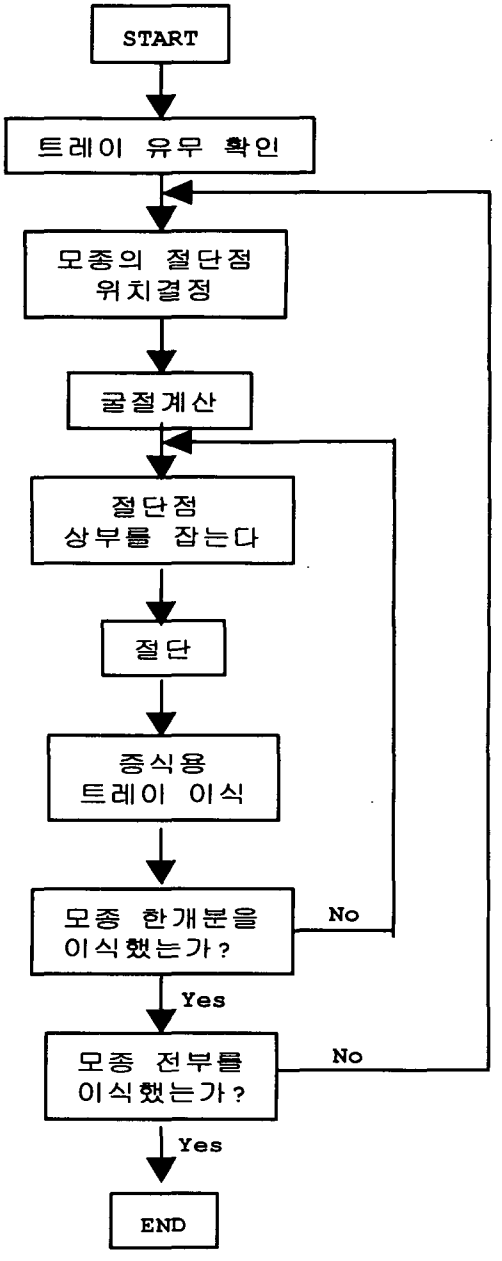


그림 1.7 로봇 작업의 공정도

여기서 묘종 위치를 검출하는 센서와 연약한 묘종을 소프트하게 잡는 그리퍼가 특징이고, 묘종을 스스로 발견해서 미리 정해진 간격으로 절단해서 이식하는 시스템이다. 묘종의 절단 위치를 스스로 정하기 위해 묘종의 3차원 형상검출, 마디부의 자동 인식 기능을 갖추었다. 또, 전자동화를 위해 묘종의 자동 공급 배출이 가능하도록 해서, 실용적인 시스템로서의 기능을 모두 가지도록 한 시스템을 개발했다. 묘종은 플라스틱 트레이에서 재배시켜, 트레이 단위로 이용하였다. 트레이의 외형은 200x248 mm로 4x5=20 개소의 구멍이 있고, 여기에 섬유상의 재배지를 꽂고 묘종을 배양한다.

2 대의 트레이 스토커는 묘종 트레이와 이식용 트레이를 격납하고 있어, 필요에 따라 배출 격납한다. 트레이 스토커 사이를 모노레일 식의 컨베이어로 연결하여 트레이를 반송한다. 와이어로 구동되는 2 대의 캐리어가 하나의 레일 위를 독립적으로 움직이기 때문에 공간 절약을 도모할 수 있다는 특징을 가진다. 중앙의 트랜스퍼에 묘종 트레이를 셋팅해서 2 대의 로봇으로 작업한다. 한 쪽의 인식 로봇은 직교 좌표형으로, 선단에 묘종의 3차원 형상을 검출할 있는 헤드를 가지고 있다. 다른 한 쪽의 이식 로봇은 본체부에 직동축을 가진 6축 수직 다관절형이고, 묘종의 파지 부분과 절단하는 부분을 일체로 해서 캠팩트화를 도모한 컷터 및 그리퍼 부분을 앤드-이펙트에 부착하였다. 제어장치는 2 대의 컨트롤러로 구성되어 있다. 각 각의 컨트롤러는 복수의 32 비트 마이크로 컴퓨터를 가지고, 로봇과 주변장치를 제어하고 있다.

2) 소식물체 파지 및 절단 메카니즘

이 핸들링 로봇 시스템은 조직 배양의 증식 공정에서 이용되기 때문에 무균 상태 하에서 사용되므로 로봇에 요구되는 기본 기능을 살펴보면

다음과 같다.

① 묘종을 임의의 자세로 임의의 위치로 운반할 수 있다.
② 무균실 내에서는 부식하지 않아야 한다.
③ 멸균이 가능하여야 한다.

이 연구는 ①의 기능에 중점을 두고 일반적인 6축 수직 다관절 로봇을 사용하여 핸들링 동작의 유연성에 중점을 두어 연구하였다. 연약한 배양 묘종을 다루는 경우에 중요한 요소로서 묘종을 손상시키지 않고 잡을 수 있는 그리퍼가 있다. 쥐는 힘이 너무 강하면 묘종은 상하고 지나치게 약하면 그리퍼가 잘 미끄러져 이식하기 어렵다. 핀셋에 스트레인 게이지를 붙여 실제로 배양 묘종을 이식하는 작업을 해본 결과 40 gf 정도의 쥐는 힘이 가장 적합함을 알았다. 또, 배양 묘종을 다루는 경우, 소식물체가 작아 세세한 작업을 많이 하기 때문에 그리퍼 본체는 가능한 작은 소형이 좋다. 최근 형상기억 합금이나 판 스프링을 사용한 것을 이용한 연구 사례가 있다. 그러나, 모두가 기구부의 강성을 이용해서 쥐는 힘을 발생시키는 것이고, 잡는 물체의 센터링 강성도 스프링부에서 정한다. 또 잡는 묘종의 외형이 변하면 잡는 힘이 변해 버린다. 이 장치에서는 파지 부분에 붙은 스트레인 게이지의 정보를 피드백해서 그리퍼를 개폐하는 방식을 채용해서 힘 제어의 문제를 해결했다. 또 그리퍼부가 가능한 작아지도록 본체와 구동 부를 나누어서 구동력은 와이어 로프로 전달하는 방식을 채용했다.

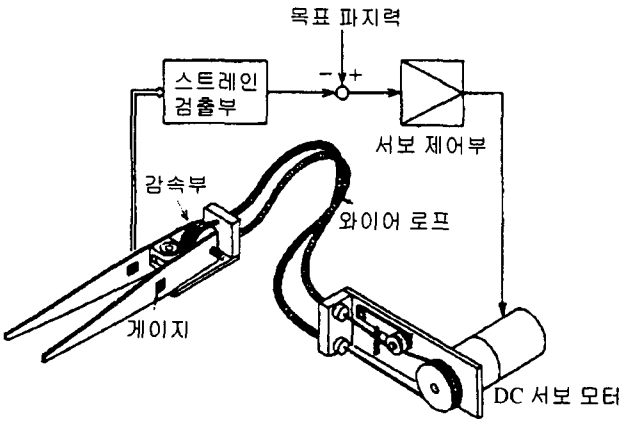


그림 1.8 Soft Gripper

그리퍼 구조는 그림 1.8에서와 같이 핀셋을 이용한 파지부와 감속부, 와이어 로프부와 모터 구동부로 이루어져 있다. 통상 이와 같은 구조에서는 쥐는 힘을 제어하려고 해도 모터에서 파지부까지의 전달계에서의 마찰과 백래쉬에 따른 비선형적 성질 때문에 제어 게인을 많이 올리지 않는 한 정도를 높일 수 없다. 이 장치에서는

① 와이어 로프와 외경 튜브를 테프론으로 코팅하고,
② 와이어 로프에 대한 파지부의 감속비를 크게 하고,
③ 파지부의 강성을 작게 (묘종의 강성 이하)한다.

이것은 다음과 같은 효과가 있다.

①은 마찰계수를 적게 한다.

②는 첫째 구동력이 적기 때문에 마찰력이 적어지고, 둘째 와이어 로프의 파지측 환산의 등가 강성이 커질 수 있고, 셋째 파지부에서 발생하는 전달계의 백래쉬를 적게 할 수 있다.

③은 대상 묘종의 단단함이 제어계에 거의 영향을 미치지 않으므로

일정한 서브게인으로 항상 안정하게 잡는다.

소프트 그리퍼의 구성을 표 1에 나타낸다.

표 1.1 소프트 그리퍼 사양

항 목	사용 방법
제어방식	파지력 피드백 방식
구동방식	DC서보모터+감속력+와이어
파지력	0~200 gf
정밀도	3 gf
응답주파수	3 Hz

소식물체 절단에는 메스 형상의 칼날류나 가위가 필요하다. 이 장치에는 일부 해부용 가위를 유용해서 묘종 검출부의 선단에 설치했다. 묘종 검출시에 방해가 되지 않도록 그림 1.9의 한 점 쇄선의 위치에 격납 할 수 있는 구조로 하고 또 와이어 로프로 다른 장소에서 구동할 수 있도록 해서 본체부를 작게 하였다.

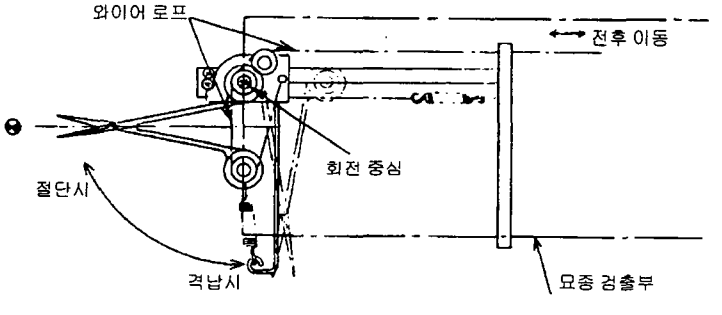


그림 1.9 커터부의 구조

4) 소식물체 절단위치 인식 시스템

일본 도시바 생산기술연구소에서 소식물체를 일주씩 직립으로 생육하는 경우에 대해 묘종 인식 및 절단 위치 검출 방법에 대해 연구를 하였다. 이 경우 묘종의 위치 정보를 알기 위해서는 묘종에 대한 3차원 정보가 필요하다.

실제 묘종은 일반 공업제품과 달리 모양이 일정하지 않기 때문에 어떤 방법으로 묘종의 형상을 검출해서 절단하여야 할 위치와 잡아야 할 위치를 계산하지 않으면 안 된다. 이 부분이 묘종 검출부이다. 현 상태에서는 묘종의 잎을 보지 않고 기계적으로 3 분할해서 절단해야 할 위치를 검출하고 있다. 검출법은 범용성, 외란의 제거, 고속성을 고려하여 그림 1.10과 같이 묘종 방향으로 레이저 스포트광을 스캔해서 그 광을 다른 방향에서 광위치 검출기(PSD)로 보는 것으로 반사된 위치의 3 차원 좌표를 계산하는 삼각측량 방식을 채용했다.

묘종 검출부는 그림 1.11과 같이 PSD 카메라부, 레이저 발광부, 레이저 스캔너부(이후 스캔너라고 부른다) 및 이들 전체를 좌우, 상하, 전후로 움직이는 구동부로 이루어진다. 스캔너부는 칼바노 미러로 레이저를 수평으로 스캔한다. 검출은 우선 장치를 묘종이 있을 만한 방향으로 향하게 두고 레이저 스포트광을 수평으로 스캔해서 광이 반사하면 거기에 묘종이 있다고 해서 수차 그 3차원 좌표를 계산해서 기억한다. 1 스캔 분으로 기억한 위치 데이터를 평균해서 줄기의 중심 좌표로 하고 있다. 실제는 레이저의 초점 심도가 작은 것, 삼각측량 계산을 선형화 해서 간략화 한 것, PSD 카메라의 켈리브레이션을 생략한 것 등의 이유로 PSD 카메라 중심부에 스포트 광이 닿지 않으면 정밀도가 나오지 않는다. 그래서 2회 묘종의 위치를 검출하려고 하고 있다. 우선 어느 일정 위치, 자세로 묘종을 검출해서 대략적인 위치를 계산해서, 그 데이터의 초점과 선형이 가장 좋아지도록 위치, 자세를 수정해서 다시 한번 검출하는 2 스텝으로 가능한 한 정확한 위

치를 계측하도록 한다. 또, 상중하의 3점을 검출해서 줄기가 넘어지는 것과 굽는 것을 계산해서 절단점과 잡는 점의 위치와 자세를 산출해서 로봇 본체에 지령하도록 하고 있다.

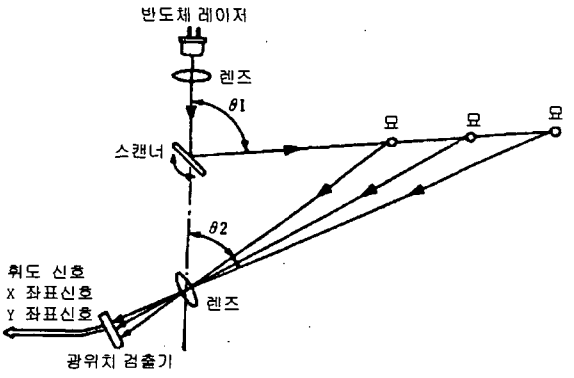


그림 1.10 묘 위치 계측의 기본원리

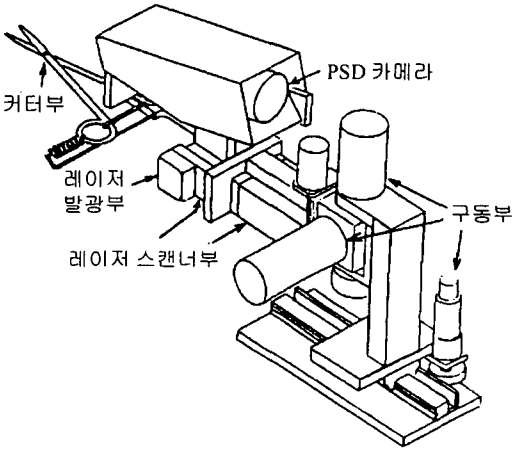


그림 1.11 검출부의 구조

묘종의 측방에서 레이저 센서로 비접촉식으로 검출해서 절단 위치를 결정한다. 묘종의 대략적인 위치를 입력하는 것만으로 후방이나 측방 묘종의 영향 없이 대상으로 하는 묘종의 절단 위치를 결정할 수 있는 특징을 갖는다. 이 검출 부분은 인식 로봇의 앤드-이펙트부에 설치되어 있으며 그림 1.12와 같은 구조를 하고 있다. 레이저광을 칼바노 미러에서 수평으로 주사해서 묘종의 수평 단면을 검출할 수 있다. 높이를 바꾸어서 측정하는 것으로 묘종 전체의 3차원 형상을 확보할 수 있다.

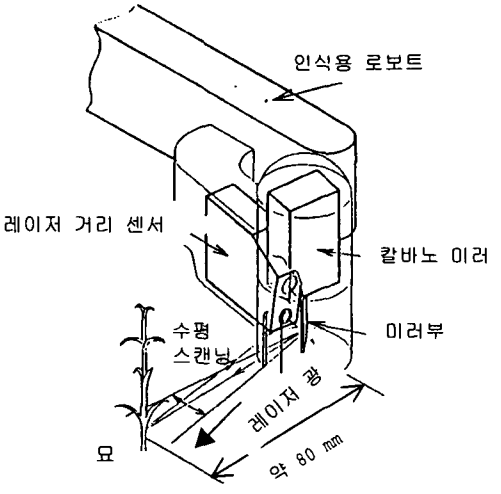


그림 1.12 묘 3차원 형상 계측부

인식은 그림 1.13과 같은 순서로 한다. 레이저광을 스캔하는 방법은 검출하는 영역의 넓이에 비례해서 시간이 걸리기 때문에 검출을 조검출과 정밀검출 2 단계로 해서 고속화하였다. 우선 조검출에서 묘종이 있어야 할 넓은 범위에서 레이저광을 주사해서 묘종의 유무 및 단면을 확보한다. 내부 방향에 대해서 일정 범위 이외의 데이터를 제거해서 굵은 부분 하나만 보일 경우 줄기라 판단해서 묘종의 위치와 줄기가 자라고 있는 방향을

결정한다. 다음에 정밀검출을 행하고, 정밀검출의 정보를 사용해서 줄기 부근을 위 방향으로 거슬러 올라가서 좁은 폭의 상세한 형상을 검출한다. 얻어진 형상에서 줄기의 굵기, 분기의 유무 등의 룰에서 마디 부분을 인식한다. 마디와 마디 사이의 협소한 줄기 부분을 절단한다. 절단위치는 다음 방식에 따라 산출한다.

① $L \geq 6$ 일 때

$x = 3.5 + 0.4(L-6)$ mm

② $L \leq 6$ 일 때 절단하지 않는다.

L : 마디와 마디 사이의 줄기 길이

x : 절단위치(줄기 하단에서의 거리)

이것은 L이 긴 경우는 상하 줄기의 길이를 적절하게 하기 위해서이다. 짧은 경우는 위 마디의 잎과 아래 마디 잎 사이에 커터 그리퍼가 들어가지 않기 때문에 절단하지 않는다. 인식한 결과의 데이터는 절단위치 및 절단각도를 나타내고 있다.

시스템에서 이식용 묘종을 심을 수 있는 트레이가 공급되면 그것을 검토해서 묘종을 보고 잘라 증식용 트레이에 이식 작업을 시작한다. 묘종의 유무를 체크하면서 일련의 이식 작업을 하게 되는데 작업시간은 일절편당 약 20초 가량 걸리는 것으로 나타났다.

지금까지 일본 도시바 생산 기술연구소에서 개발한 시스템에 대해 알아보았다. 이 시스템은 앞서 설명한 필요기능 중 아직 일부분의 개발한 것에 지나지 않고 아직 많은 부분에 대해 계속적으로 연구개발이 필요한 것으로 나타났다.

우선 가장 필요한 것이 재배 묘종을 절단하는 위치를 인식하는 S/W이다. 이것은 소식물체의 종류에 따라 각각 다르고, 마디 사이를 절단

해서 좋은 것, 포기 나눔이 필요한 것, 구근상인 것 등, 여러 가지로 다양하다. 이중에서 실제 적용하고자하는 것을 선정해서 개발하지 않으면 안된다. 포기 나눔 S/W 등은 매우 중요한 S/W이지만 거의 불가능에 가까울 만큼 어려운 S/W이다.

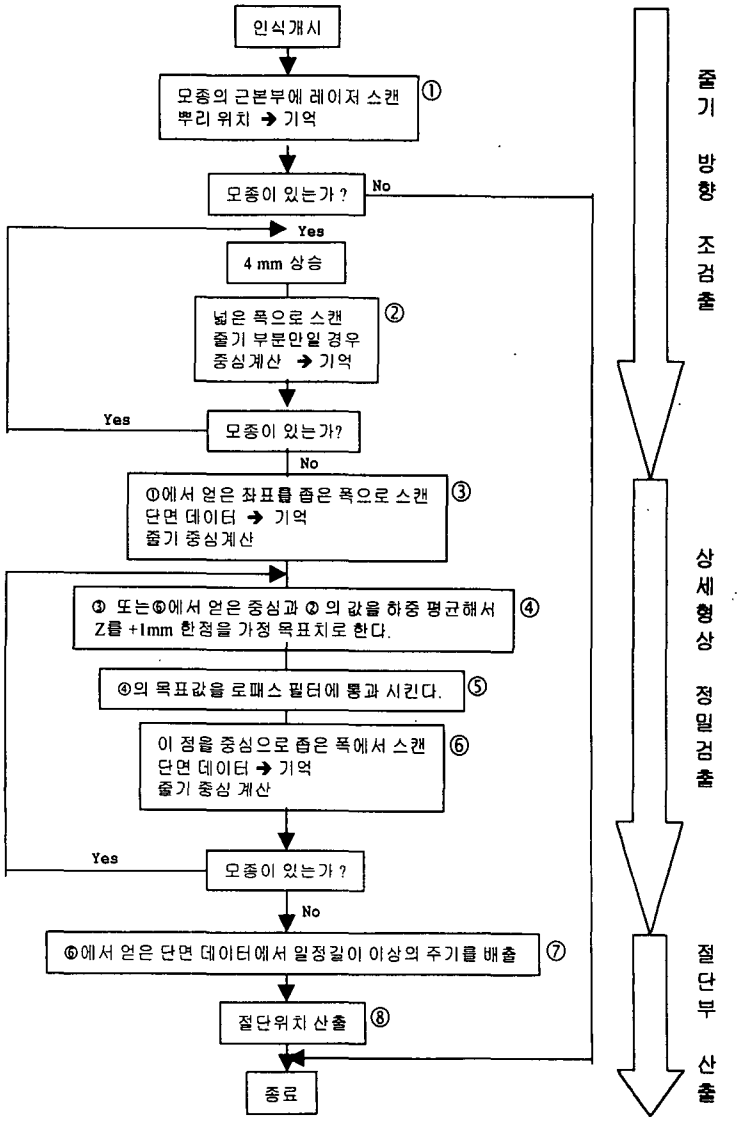


그림 1.13 묘종 검출 알고리즘 순서

도시바 연구소 실험실에서 개발한 시스템은 일절편을 절단해서 이식하기까지에 걸리는 시간이 약 20초이다. 이것은 작업 속도 측면에 대해서는 그다지 고려하지 않고 개발했기 때문이다. 실용화를 위해서는 절단 위치를 인식해서 이식하기까지의 소요시간을 5초 이내로 하여야 한다. 소식물체에 따라서는 2초 정도를 요구하는 것도 있을 수 있다.

멸균의 가장 좋은 방법은 열에 의한 살균이지만 로봇 전체를 가열해서 살균하는 것은 곤란하다. 이것도 묘종을 배양하는 시스템 전체 안에서 고려해야 하는데, 로봇 전체는 아니고 파지 및 절단하는 부분만 살균하는 방법으로 검토를 진행해 가지 않으면 안 된다.

제 4 절 조직배양실 자동화를 위한 배양방법 및 배양용기

1. 조직배양실의 씨감자 배양방법

조직 배양실의 증식공정을 자동화하기 위해 현재 수작업 공정으로 하고 있는 배양실 증식 작업과 배양 방법에 대해 알아본다. 그림 1.14와 같이 특수 제작된 페트리 디시에서 씨감자 줄기를 배양한 것을 보여준다. 이곳에서는 배양 줄기를 랜덤한 형태로 투입(이식)하여 배양하기 때문에 씨감자가 랜덤한 방향으로 자라서 자동화하기에 많은 어려운 점이 있다.

씨감자 증식 작업은 성숙한 씨감자 줄기 마디를 잘라서 다시 페트리 디시에 투입(이식)하여 배양을 반복함으로서 씨감자를 증식하여 간다. 작업자가 비교적 청정 환경을 갖춘 작업실에서 배양된 씨감자 페트리 디시를 열어서 한 손에 핀셋, 다른 손에 절단 가위를 잡고 페트리 디시로부터 씨감자 줄기를 집어들어서 한 마디씩 잘라서 새로운 배양 페트리 디시에 투입하여 배양실로 옮겨 배양을 한다. 여기서 서로 엉겨서 랜덤하게 자란 씨감자 줄기들 중에서 줄기 하나를 찾아서 절단하는 과정은 인간에게는 비교적 쉬운 작업이 되지만, 이 작업 과정을 기계로 대체하기 위해서는 많은 어려운 문제에 봉착하게 된다. 인간의 지능과 팔 동작의 유연성이 갖는 기능을 기계에 부여할 수 있어야만 기계가 이 작업을 대체할 수 있게 된다. 이 수 작업에서 인간의 눈이 갖는 역할을 거의 절대적인 것이다. 페트리 디시에서 여러 줄기가 엉겨서 있는 줄기 중에서 하나의 줄기를 찾고, 집어야 할 위치를 찾고, 절단하여야 할 위치, 이식 페트리 디시에서 투입하여야 할 위치 등을 찾는데 있어서 가장 중요한 핵심적인 역할을 한다.

현재의 첨단기술 중에서 인간의 눈의 역할을 대신할 수 있는 것은

머신 비전(Machine Vision)기술이 있다. 이 기술의 현재 수준은 아직도 초보단계를 벗어나지 못하고 있으며 페트리 디시에서 랜덤하게 자란 씨감자 줄기들 중에서 줄기 하나 씩 찾아내는데 적용하기에 여러 가지 어려움이 있다. 랜덤하게 엉겨서 자란 여러 씨감자 줄기 중에서 적절한 하나의 줄기를 찾는 것은 소위 말하는 빈-피킹(Bin-Picking) 문제이다. 이 문제는 화상처리 기술에서 풀어야 할 난제 중 하나이며 현재로서는 실제 문제에 적용할 수준의 기술로까지 발전하지 못했다.

그림 1.14 페트리 디시에서 다주포복 배양의 성장 형태

일반 공장에서 생산 공정을 자동화를 하기 위해서 많은 관련 작업 환경 조건에 대해 표준화 작업이 선행적으로 이루어진다. 따라서 배양실 증식공정을 자동화하기 위해서는 증식공정 관련 작업환경을 표준화하여야 한다. 즉 배양 방법, 배양 용기, 배양실 조명 등의 작업환경에 대해 표준화

작업이 선행적으로 이루어져야 배양실에서의 작업에 대한 자동화가 용이하게 이루어질 수 있다. 식물의 성장을 표준화 하기는 어렵지만 자동화에 필요한 최소한의 표준화된 생육 방법을 고려한다. 배양실 조건을 자동화에 적합한 성장을 할 수 있도록 하여야 한다. 예를 들면, 배양실에서 광조명은 씨감자가 성장하는 방향에 설치하는 것이 가장 좋다.

그림 1.14는 씨감자의 다주 포복배양을 하였을 성장 형태를 보여주고, 그림 1.15는 일주 포복 배양을 하였을 때 성장 형태를 보여준다. 넓은 둥근 페트리 디시에서 일주 포복 배양은 랜덤한 방향으로 자라나간다. 이렇게 자란 형태에서 씨감자의 잎이 서로 엉겨서 줄기와 잎의 관계가 명확하지 않으며 마디와 마디 사이의 간격을 구별하기 어렵다.

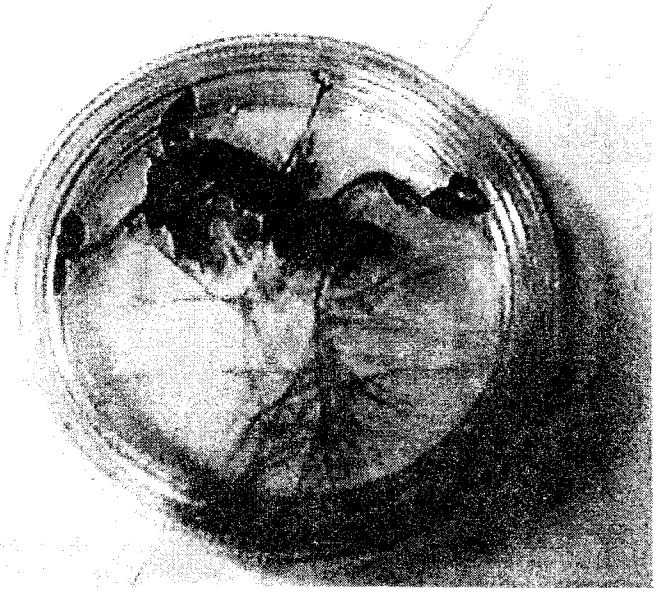


그림 1.15 씨감자 일주포복 배양의 성장형태

이 배양 방법에서 배양 용기를 일정한 형태로 하면 배양 용기 형태에 따라 씨감자 줄기가 성장을 하게 된다. 여기에 배양실의 광조명을 성장하는 방향에 설치함으로서 성장 방향을 어느 정도 일정하게 할 수 있다. 일주씩

배양을 하게 되면 사용하는 배양 용기의 개수가 많아지는 단점을 갖게 된다. 배양실에서 용기가 차지하는 면적이 많아지고, 용기를 취급하는 횟수가 많아진다. 배양실의 단위 면적당 생산량이 줄어들 수가 있고, 또한 생산비의 증가가 있을 수 있다.

그림 1.16 일주포복 및 일주직립 배양 방법

그림 1.16은 일주직립과 일주포복 배양 방법에 대해 실험 배양을 하였다. 일주직립으로 성장한 씨감자에 대해서는 자동화 하기가 비교적 쉽다. 이것은 씨감자가 한 줄기씩 일정한 방향으로 자라고 성장 형태가 비교적 정형적으로 되기 때문이다. 여기서는 적어도 빈피킹과 같은 문제가 발생하지 않는다. 실험 결과 씨감자는 배양용기의 형태에 따라 성장은 하지만, 그림 1.17, 1.18에서 보는 바와 같이 성장에 있어서 씨감자 줄기가 튼튼하지 못하고 줄기 자체가 퇴화되어 가는 현상이 발생하였다. 직립 배양에서 얻은 줄기는 튼튼하지 못해 증식용으로 반복하여 사용하기 부적합하므

로 씨감자 증식 배양에서 직립 배양은 부적합하다. 그러므로 씨감자 배양은 포복 배양 방법으로 배양을 할 수 밖에 없다.

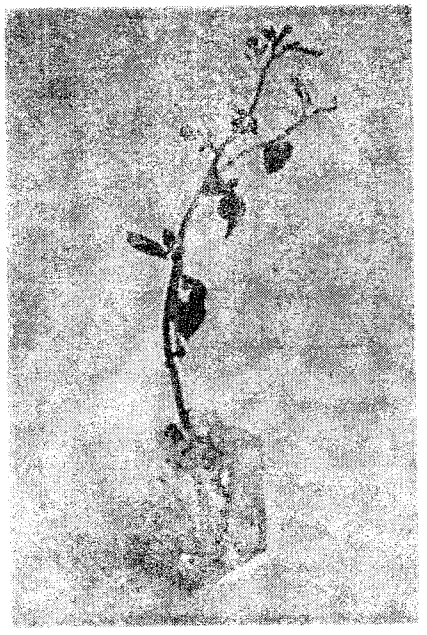


그림 1.17 일주 직립재배

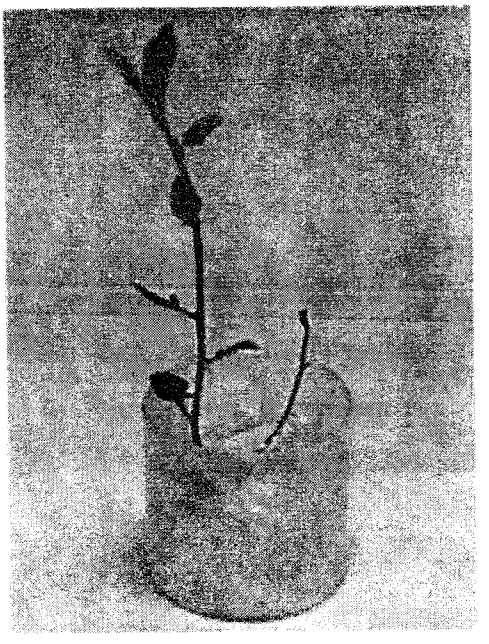


그림 1.18 일주 직립재배

그림 1.19, 1.20에서 보는 바와 같이 일주포복 배양을 좁은 긴 직육면체 배양 용기를 만들어 배양을 한 결과 자동화하기에 비교적 좋은 성장 형태의 결과를 얻었다. 여기서 자동화하기에 적합한 배양용기의 크기와 씨감자의 성장을 비교적 일정하게 할 수 있는 생육 환경을 조성해주면 될 것이다. 특히 광조명은 성장 방향에 설치하여야 한다.

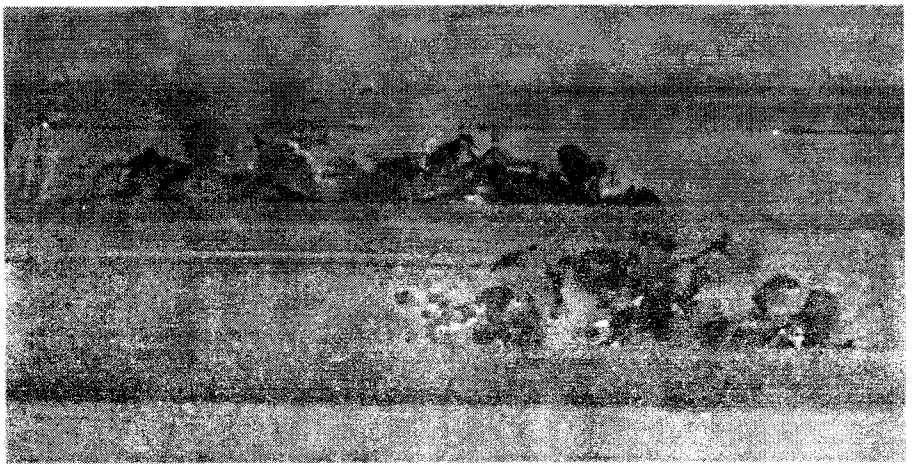


그림 1.19 일주 포복재배

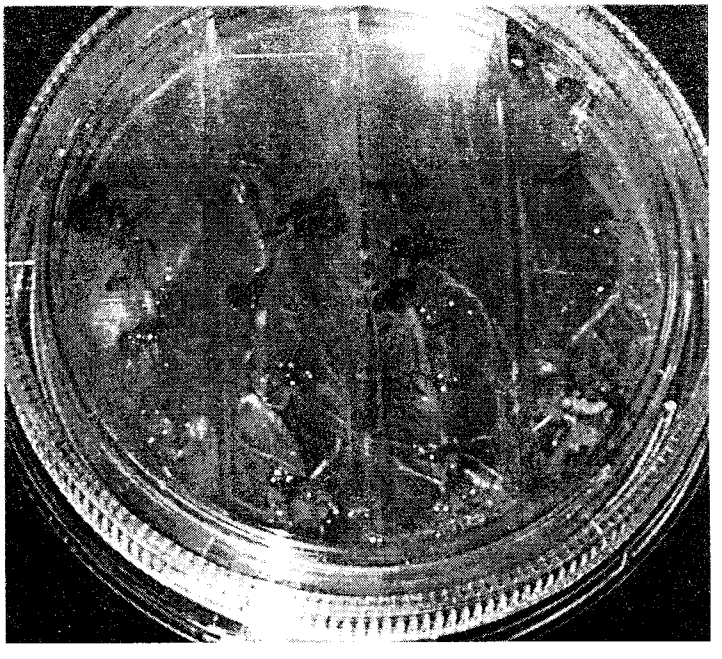


그림 1.20 일주 포복재배

2. 조직배양실의 씨감자 배양용기

기존 원형 페트리 디시에 씨감자를 재배하면 자유형태로 성장하기 때문에 기계화 및 자동화하기에 많은 어려움이 있다. 그래서 기존 원형 페트리 디시에 그림 1.21과 같은 아크릴 판을 이용하여 페트리 디시에 내부 칸막이를 만들어 넣어서 씨감자 종묘가 직선으로 성장하도록 하였다. 현재까지 성장 과정을 살펴 본 결과로는 비교적 직선 형태로 잘 성장하지만 생물의 특성상 광원 방향으로 향하여 성장하기 때문에 배양시 광조명의 위치는 씨감자 성장 방향에 설치하여야 한다. 이 때 광조명의 밝기는 성장의 속도와 관계가 되기 때문에 배양 시스템 구성시 충분히 고려가 되어야 한다.

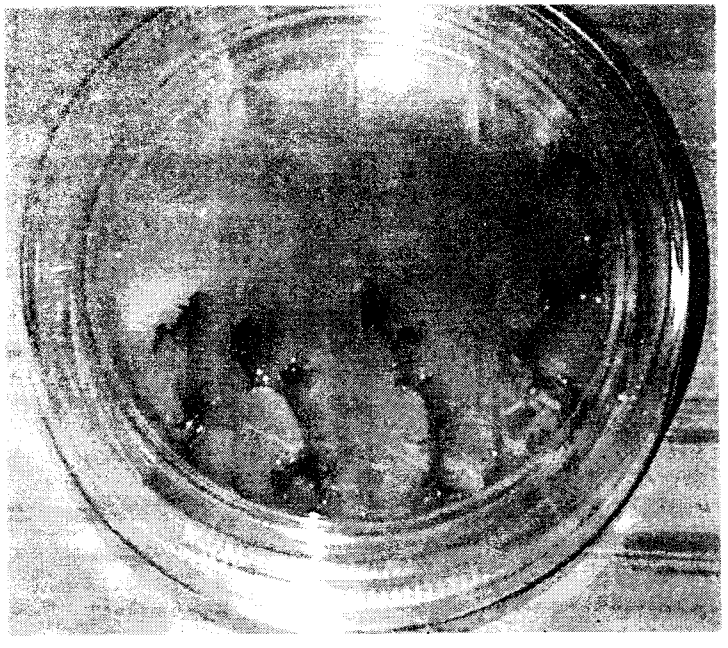
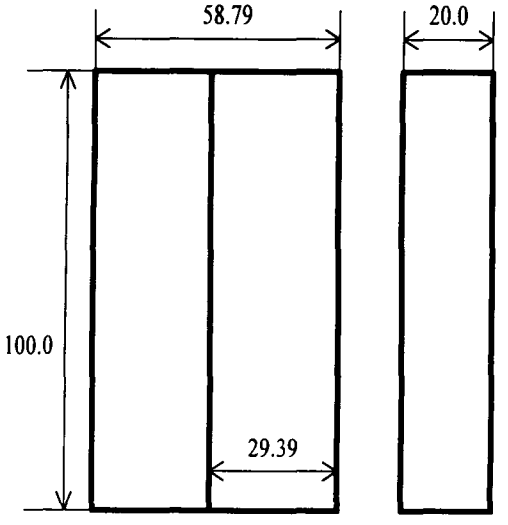


그림 1.21 페트리 디시 칸막이에서 씨감자 성장형태



샬레 폭 = 29.39
샬레 길이 = 100.0
샬레 높이 = 20.0

그림 1.22 실험 배양용 페트리 디시 칸막이

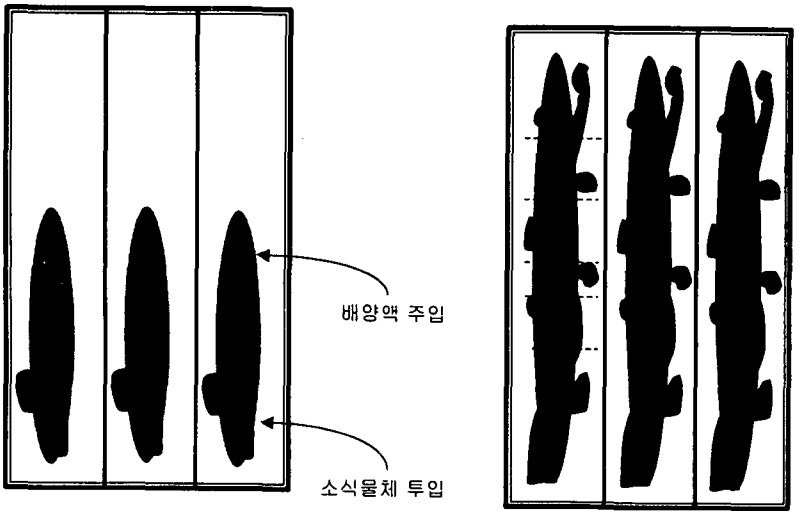


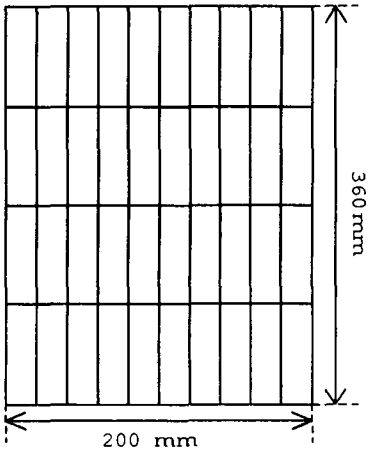
그림 1.23 실험 배양용 디시 칸막이 내에서 배양 방법

3. 조직배양실의 작업환경 표준화

조직 배양실 작업환경의 특성을 고려하여 수 작업의 증식공정을 기계화 및 자동화를 하여야 한다. 증식 공정의 주요 작업환경은 표 1.2와 같으며, 본 연구에 가장 적합한 배양방법은 일주포복 배양방법이다. 배양 용기의 형상은 일정하게 자라게 할 수 있는 형상을 갖는 것이 가장 적합하다. 현재 사용하는 용기 페트리 디시는 뚜껑의 개폐 동작을 자동화하기에 약간의 문제가 있으나 자동화에는 큰 지장이 없다. 본 연구에서는 사용하지 않았지만, 본 연구 결과로서 제안한 배양 용기의 형상은 그림 1.24와 같으며 직사각형 형상의 멀티-셀을 갖는 트레이 형상이 자동화에 가장 적합하다. 직사각형 셀 모양은 20x90x20 mm의 형상 치수을 가지며, 4x10 개의 셀을 갖는 트레이 형상이다. 트레이의 뚜껑 개폐는 상하 동작에 의해 이루어지도록 하였기 때문에 뚜껑의 개폐 동작에 대해 자동화하기에 쉬운 구조이다.

표 1.2 조직 배양실 작업환경

항 목			특 성
배양 방법	다주	포복배양	▪ 랜덤한 방향으로 성장 ▪ 수작업, 자동화 불가 ▪ 성장이 튼튼함
		직립배양	▪ 일정한 방향으로 성장 ▪ 자동화 비교적 쉬움 ▪ 성장이 퇴화됨
	일주	포복배양	▪ 랜덤한 혹은 광조명 방향으로 성장 ▪ 자동화에 제약이 있음 ▪ 성장이 튼튼함
		직립배양	▪ 일정한 방향으로 성장 ▪ 자동화가 가장 쉬움 ▪ 성장이 퇴화됨
배양 용기	원형 페트리 디시		▪ 자동화에 부적합함 ▪ 뚜껑 개폐 불편
	사각형 트레이		▪ 자동화에 적합함 ▪ 뚜껑 개폐 용이
광조명			▪ 성장 방향에 설치 ▪ 광 투과성을 고려한 배양용기 뚜께



트레이 셀 폭 = 20 mm
트레이 셀 길이 = 90 mm
트레이 셀 깊이 = 20 mm

그림 1.24 배양 트레이 형상

제 2 장 조직배양실 소식물체 인식 시스템

제 1 절 화상처리 및 조명 장치

1. 화상처리 시스템

본 연구개발에 이용한 비전 시스템은 Cognex VP50 이며, 이것은 자동 정열, 조립, 검사 등의 반도체 및 전자부품 생산 공정에 많이 활용되고 있는 비교적 저가의 Grey-Scale Machine Vision 시스템이다. 이 시스템의 특징으로서는 "Normalized Correlation Search"이라 불리는 Grey Scale Template Matching Algorithm으로 간단하거나 복잡한 것을 가리지 않고 어떤 목표 대상을 찾아낸다. 이 알고리즘은 찾고자하는 목표 대상을 미리 정의된 Grey-Level의 모델과 일치하는지 이미지를 스캔닝하여 판정한다. Black이나 White로 이미지 픽셀의 값을 떨어뜨리는 간단한 Binary Vision System과는 달리 Search는 Pattern을 찾아내고 위치를 측정하기 위한 더 많은 이미지 데이터를 제공하는 256 Grey-Scale 값으로 작동한다. 이런 충분한 정보 데이터를 가지고 조명이나 공정 또는 표면 변화와 같은 외관의 변화에도 불구하고 신뢰성을 갖고 반복적으로 Pattern의 위치를 찾아낸다. 또한 이 시스템은 오전 오후의 광량 변화 혹은 현장 조명의 변화에 잘 적응하여 신뢰성 있는 결과를 도출할 수 있다. 이미지의 조도나 명암 변화에 무관한 기능은 Binary Vision System으로는 할 수 없는 것이다. 이 시스템의 처리 속도는 적용 모델의 크기, Search 범위, 사용 중인 호스트 컴퓨터의 속도 등과 같은 변수에 따라 달라지지만 대부분의 모델에 대해 수 ms 이내에서 Pattern 위치를 찾아낸다. 응용 분야에 따라 달라지

지만 목표 위치를 1/4에서 1/10 서브 픽셀의 단위로 찾아 낼 수 있다. 이 시스템은 ISA Bus를 채용하고 있고 1024x512 Pixel Frame Memory를 이용하여 4대의 RS-170이나 CCIR 카메라로부터 Video Input를 받아들인다. 보드 상의 이미지를 실시간으로 VGA 모니터에 디스플레이하는 회로가 장착되어 있다. 이것은 획득한 이미지를 VGA 모니터로 Pass-Through 해주기 위한 Feature Connect 인터페이스를 지원하는 VGA 모니터 컨트롤러가 필요하고 별도의 비전 시스템 모니터는 필요가 없다.

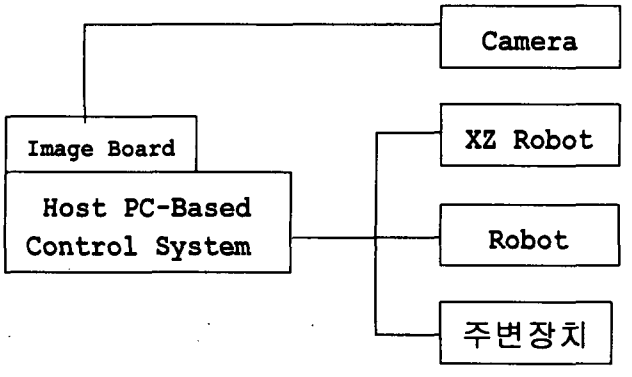


그림 2.1 화상처리 및 컨트롤 시스템 구성

- Cognex VP52
- Borland C++
- Pentium PC
- Camera : Color and B/W
- Lens : 25 mm
- OS : Window98

2. 조명 시스템

화상처리 비전 시스템에서 조명의 역할 매우 중요하며, 조명에 따라 화상처리 난이도가 달라지기 때문에 작업환경에 가장 적합한 조명을 사용하여야 한다. 본 연구에서는 처음에 작업암실을 이용하여 실내 조명에 관계없이 일정한 조명을 줄 수 있는 시스템을 구성하였다. 이 작업 암실을 이용하여 화상처리를 한 결과 화상처리는 비교적 좋은 결과를 얻었으나 이 작업 암실 자체를 운영하는데 여러 가지 복잡한 구동 시스템이 필요하게 되었다. 이 작업 암실을 사용하지 않고 일반 실내 조명을 이용할 경우 조명의 변화가 심하기 때문에 화상처리에 많은 어려움이 발생하였다. 이 작업 환경을 재검토한 결과, 배양용기가 빛을 통과시키는 투명한 재료 되어 있기 때문에 백라이팅(Back Lighting) 방법을 사용하였다. 이 방법은 외부 조명 변화에 대해 비교적 민감도가 덜하기 때문에 별도의 암실이 필요 없다. Structured Lighting 방법을 이용하면 3차원 정보를 얻을 수 있으나, 본 연구에서는 3차원 정보가 있으면 좋으나 2차원 정보만 가지고도 충분히 작업이 가능하기 때문에 Structured Lighting을 사용하지 않았다.

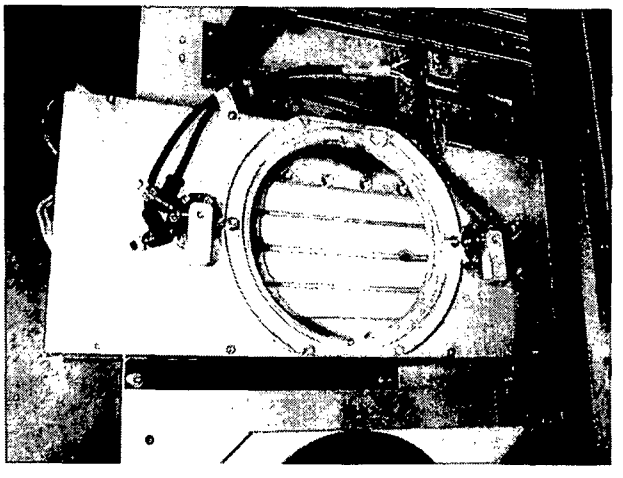


그림 2.2 Back Lighting 조명장치

제 2 절 소식물체 파지 및 절단 위치결정 알고리즘

1. 스캔 및 체인 코딩 방법

주사선 방향으로 탐색하다가 배경에서 대상물로 변환되는 픽셀를 만나면 그 픽셀을 시작 픽셀로 하여 해당 대상물의 모든 윤곽 픽셀에 대한 체인 코드를 구한 다음 대상물을 위해 주사방향으로 탐색을 계속하는 방법이다. 주어진 화상 내에 있는 대상물 윤곽 중에서 한 화소로부터 시계 방향 또는 반시계 방향으로 물체 윤곽에 대한 체인 코딩을 수행한다면 한 대상물에 대한 윤곽 정보를 일련적으로 얻을 수 있기 때문에 데이터 관리가 용이하다. 여러 개의 대상물이 포함되어 있을 때 탐색이 완료된 대상물에 대해서는 탐색하지 않는다.

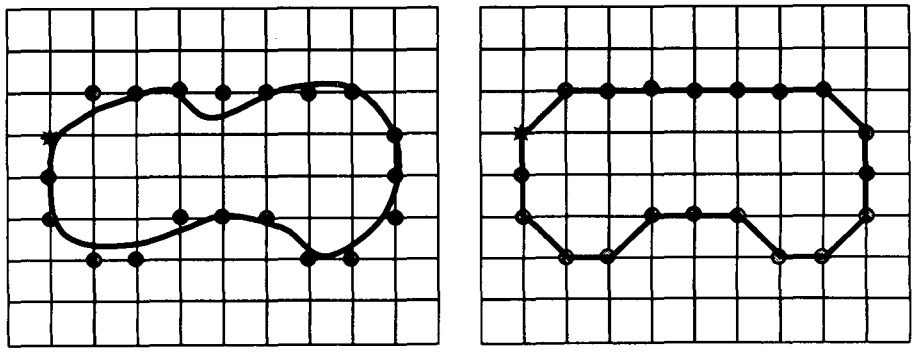
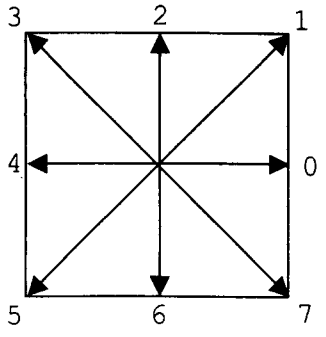


그림 2.3 그리드를 가진 원래 곡선과 정량화 점의 교차(a), 정량화 점으로 연결된 링크(b)



10000007665434454322

그림 2.4 앤코딩 스킴과 곡선의 체인 코드

2. 알고리즘 내용

처음에는 주사선 방향으로 진행하면서 배경에서 대상물로 변환되는 좌표를 조사한다. 일단 변환점을 만나면 그 점의 좌표값을 저장하고 해당 대상물의 윤곽에 대한 체인 코딩을 수행한다. 체인코딩과 함께 윤곽 픽셀의 픽셀값으로 변환한다. 체인 코딩이 끝나면 다음 물체를 만나기 위해 주사선 방향으로 진행한다.

① 주사선 방향으로 진행하면서 배경에서 대상물로 변환되는 좌표를 찾는다.

② 변환점을 만나면 시작 좌표값을 저장하고 해당 대상물의 윤곽에 대해 반시계 방향으로 체인 코딩을 수행한다. 하나의 선분을 만든다. 체인 코딩과 동시에 윤곽 픽셀의 픽셀값을 특수값으로 변환한다.

③ 새로운 변환점을 만나면 새로운 시작 좌표값을 저장하고 해당 대상물에 대해 반시계 방향으로 체인 코딩을 수행한다. 또 다른 선분을 만든다.

④ 두 선분이 만나면 교차점 좌표를 저장한다. 삼지의 형상을 탐색한다.

⑤ 특수 픽셀값을 만나면 탐색 완료된 대상물이므로 배경으로 취급하고 탐색을 계속한다.

⑥ 삼지점(Node)으로부터 일정한 거리에 파지, 절단 위치를 결정한다.

입력 항목은 배경과 대상물을 구분하기 위한 경계값(Threshold)이며, 출력 항목은 각 가지(잎)의 시작 좌표, 끝 좌표, 교차점 좌표, 삼지 방향 등이다.

소식물체 인식용 화상처리 기술에 적용한 컨버지 알고리즘은 잎과 줄기를 인식하여 구별할 수 있는 기능을 가진 알고리즘이다. 이것은 그림 2.5에서 상단 좌측에서부터 우측으로 스캔을 시작하여 세그먼트 f1의 한점을 만나면 세그먼트 f1의 연결성을 조사 판별하면서 추적을 계속한다. 좌우 스캔을 상단으로부터 하단 방향으로 스캔을 계속하여 다른 세그먼트 f2를 만나면 이 세그먼트에 대해 연결성을 계속 추적을 하게 되면 세그먼트 f1과 f2가 만나게 된다. 이 만나는 점이 마디가 되고 이 마디 다음 부분의 세그먼트는 줄기가 되고, 계속 추적하여 다른 새로운 세그먼트를 만나면 이 새로운 세그먼트는 줄기가 아니라 새로운 잎이 된다. 이렇게 하여 잎과 줄기의 식별이 된 후 다음과 같은 필요한 정보를 추출한다.

① 절단 위치결정을 위해 먼저 식물체의 줄기와 잎을 구별한 후 줄기에 있는 마디의 위치를 찾아서 그 마디간의 거리를 결정한다.

② 그리퍼 핑거 및 커터의 물리적 크기와 마디간 거리의 크기를 고려한 후 그리핑 위치와 절단 위치는 자동적으로 결정된다.

③ 잎과 줄기의 세그먼트의 시점, 종점의 좌표값과 그리핑, 절단 위치 좌표값을 로봇 컨트롤러로 보낸다.

본 연구에서는 소식물체의 줄기에 있는 마디들을 찾아서 그 마디 사이를 절단점으로 하고 절단점에서 일정한 거리의 점을 파지점으로 한다. 먼저 잎과 줄기에 관계없이 화상면에서 나누어지 않고 하나로 연결된 이미지 세그멘트를 Branch로 정의한다. 두 개의 Branch가 만나는 점이 찾는 마디점이 된다. 혹은 하나의 Branch에서 두 개의 Branch로 나누어지는 점이 마디점이 된다.

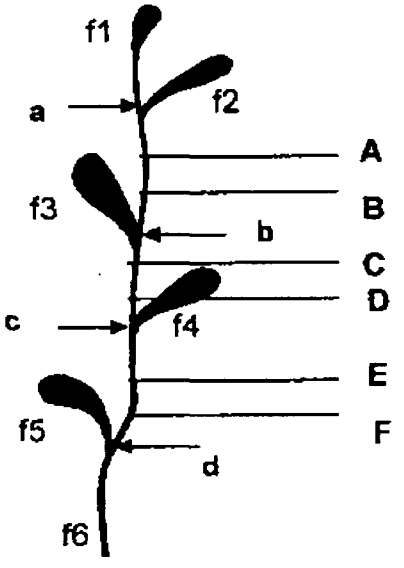


그림 2.5 인식용 소식물체 모델

■ 필요 정보

- Node Point : a, b, c, d
- Gripping Point : A, C, E
- Cutting Point : B, D, F

- 잎과 줄기의 구별
- 마디의 좌표값 (X_i,Y_i)
- 마디 사이의 거리 d_i
- ab, bc, cd, 사이 거리 l_i
- Segment 의 시점, 종점, 중앙점 좌표값

3. Branch Finder Program 구조

입력으로 Raw Data Format (*.img) 으로 크기가 512*480인 그림 파일을 읽어들여 분기가 되는 지점을 기준으로 잎이 쳐진 부분들(Segment)을 분리해 낸다.

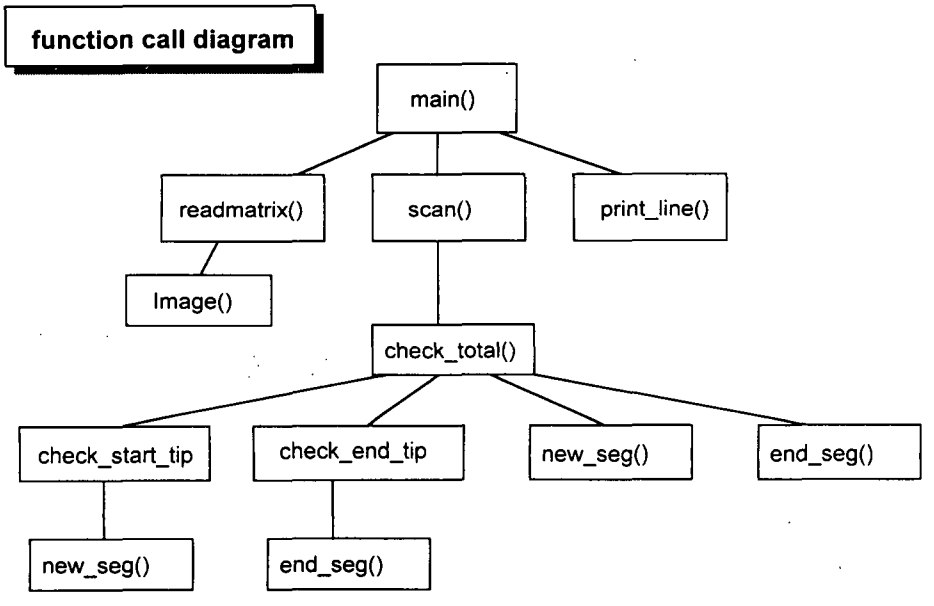


그림 2.6 프로그램 구성

컴파일하기 위해 필요한 파일들과 구성 함수들은 다음과 같다.

linemain.c
- main()
line_io.c
- readmatrix(), print_line()
scan.c
- scan.c
- skip_blank()
ch_stip.c
- check_start_tip()
ch_etip.c
- check_end_tip()
check_t.c
- check_total()
- serch_pcp()
- check_branch()
new_seg.c
- new_seg()
end_seg.c
- end_seg()
memory.c
- Image()
- Imageint()
- Imagef()
line.h
farmem.h
bf.prj

4. Branch Finder Program의 함수

(1) void main(int argc, char *argv[])

① readmatrix()를 호출함

: 입력으로 들어온 이미지 파일을 '0'과 '1'의 텍스트화일로 변환함

② 출력 결과물을 저장하기 위해 파일 이름을 하나 받아들여 Open함

③ y좌표값을 0에서 HEIGHT 까지 변화시키면서 매 루프마다 scan()을 호출함

: scan() 은 1 라인 씩 검색함

: 변수 seg_ptr은 현재 검색하고 있는 세그먼트를 가리킴

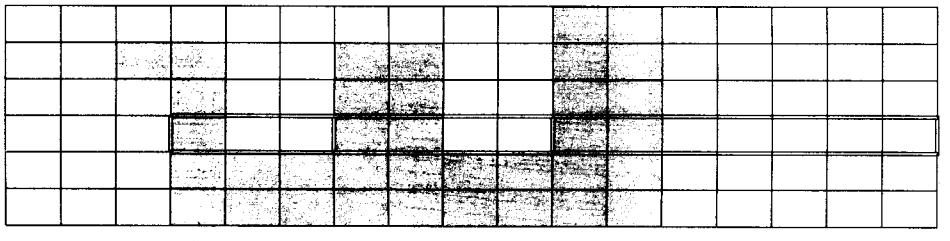


그림 2.6 그래픽 이미지 파일

④ Cutting Point 계산

⑤ 출력 파일에 결과를 기록

seg_id	StartX	StartY	EndX	EndY	Type	CutPX	CutPY
0							
1							
2							

그림 2.7 결과 저장 파일

⑥ 화면상에 결과물 출력 (print_line() call)

Seg_id	StartX	StartY	EndX	EndY	Type	CutPx	CutPY
0	352	0	273	127	200	301	63
1	258	2	266	127	200	251	64
2	407	111	335	283	200	374	197
3	270	128	297	207	100	277	167
4	187	144	282	207	200	200	175

그림 2.8 결과물 출력 화면

⑦ 할당된 메모리 해제

(2) int readmatrix(char *argv[])

① 입력화일을 처리하기위한 메모리 할당 (Image() call)

② 입력화일 개방

③ Pixel Value를 '0' 과 '1'로 변환

- Pixel Value 가 255이면 '0'으로 저장.
- Pixel Value 가 0 이면 '1'로 저장
- 첫 Row에서 처음 4개의 Value는 '0'으로 변환
- 마지막 Row는 모두 '0'으로 변환

0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0
0	0	1	1	0	0	1	1	0	0	1	1	0	0	0	0	0
0	0	0	1	0	0	1	1	0	0	1	1	0	0	0	0	0
0	0	0	1	0	0	1	1	0	0	1	1	0	0	0	0	0
0	0	0	1	1	1	1	1	1	1	1	1	0	0	0	0	0
0	0	0	1	1	1	1	1	1	1	1	1	0	0	0	0	0

그림 2.9 텍스트로 변환된 파일 구조

④ Text Type으로 변환된 정보를 저장하기 위해 File Name을 하나 받아들임

⑤ 변환된 Text 정보를 저장하기 위해 File Open

⑥ 개방된 파일에 기록

(3) int scan(void)

① skip_blank() 호출함, '1'을 발견할 때까지 X 좌표 증가

② 연속되는 '1'의 개수를 Count함

- 현재 X 좌표가 각 Row의 끝을 가리키면

  check_start_tip() 혹은 check_end_tip()을 호출함

③ 다음에 연속되는 '0'의 개수를 Count함

- 현재 X 좌표가 각 Row의 끝을 가리키면

  check_start_tip() 혹은 check_end_tip()을 호출함

  check_branch()를 호출함

④ '1'의 개수와 '0'의 개수를 인자로 하여 check_total()을 호출

⑤ 현재의 X 좌표가 현재 Row의 끝에 도달할 때까지 단계 ②, ③, ④를 반복함

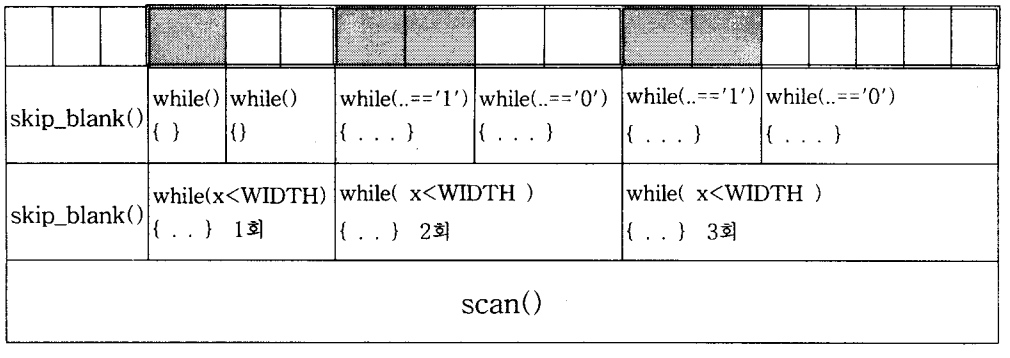


그림 2.10 scan()의 실행 구조

(4) int check_start_tip(*int ch_point, int cnt_width, int cnt_zero*)

① 잎이 쳐진 부분이 아닌 처음으로 잎이 시작되는 윗 끝 부분인지를 검사

② 윗 끝 부분이면 new_seg()를 호출하고 윗 끝부분 발견을 표시하는 Flag값을 Return 함

0	0	0	0	0	0	0	0	0	0
0	0	0 ↖	0 ↑	0 ↑	0 ↑	0 ↗	0	0	0
0	0	0←	1	1	1	→0	0	0	0
0	0	0	1	1	1	0	0	0	0
0	0	0	1	1	1	0	0	0	0
0	0	0	1	1	1	0	0	0	0
0	0	0	1	1	1	0	0	0	0

그림 2.11 check_start_tip()의 검색 구조

(5) int check_end_tip(*int ch_point, int cnt_width, int cnt_zero*)

① 잎이 쳐진 부분이 아닌 잎이 끝나는 아래 끝 부분인지를 검사

② 아래 끝 부분이면 end_seg() Call 하고 아래 끝부분 발견을 표시하는 Flag값을 Return 함

0	0	0	1	1	1	0	0	0	0
0	0	0	1	1	1	0	0	0	0
0	0	0	1	1	1	0	0	0	0
0	0	0←	1	1	1	→0	0	0	0
0	0	↙ 0	↓ 0	↓ 0	↓ 0	↘ 0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

그림 2.12 check_end_tip()의 검색 구조

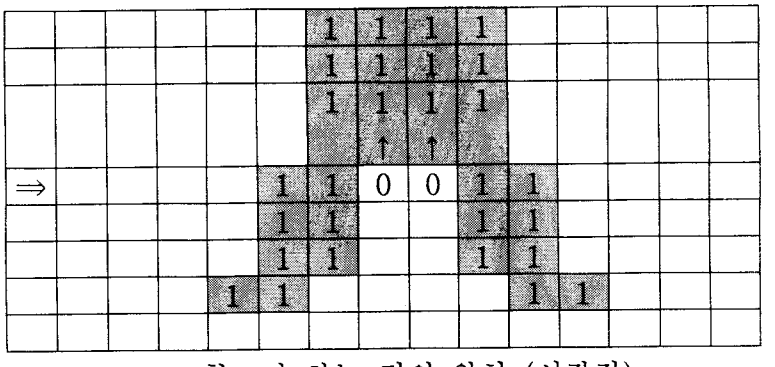
(6) void check_total(int, int, int)

① check_start_tip()을 호출

② ①의 결과, 윗 끝부분이 아닐 경우
check_end_tip()을 호출

③ ②의 결과, 아래 끝부분이 아닐 경우
분기점에 해당하는지 검사, 각 해당되는 경우에 대하여 Flag Setting

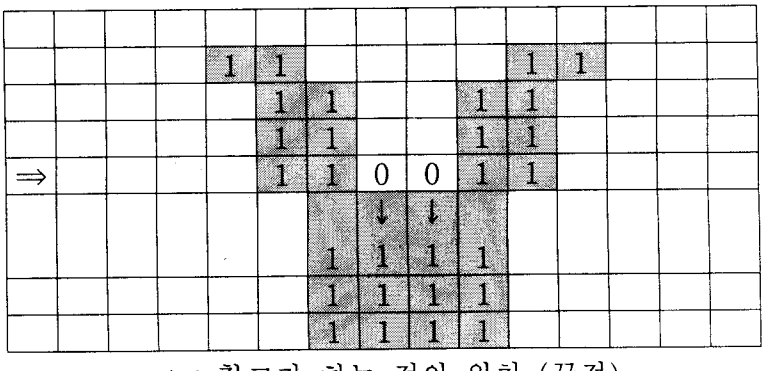
▪ Type 1 : 또 다른 세그먼트의 시작인지를 조사



⇒ : 찾고자 하는 점의 위치 (시작점)

그림 2.13 Segment Type 시작점 조사

▪ Type 2 : 기존 세그먼트의 끝인지를 조사



⇒ : 찾고자 하는 점의 위치 (끝점)

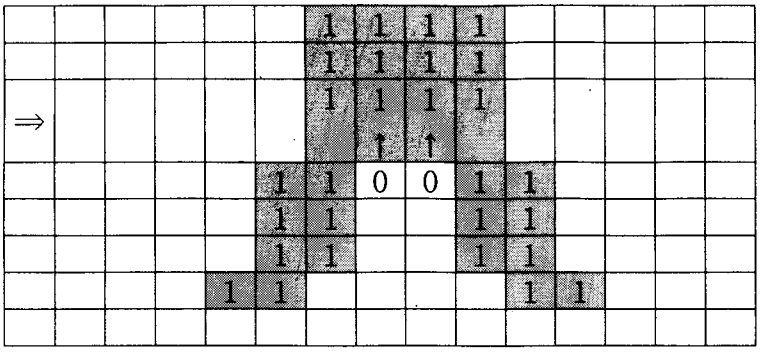
그림 2.14 Segment Type 끝점 조사

④ 각 Flag에 대하여 적절한 작업 수행.

(7) int search_pcp(int, int)

① 분기에 의해 생겨 난 앞 세그먼트의 끝을 찾음

▪ 새로운 세그먼트가 시작된 경우에 위쪽 세그먼트의 끝을 찾음



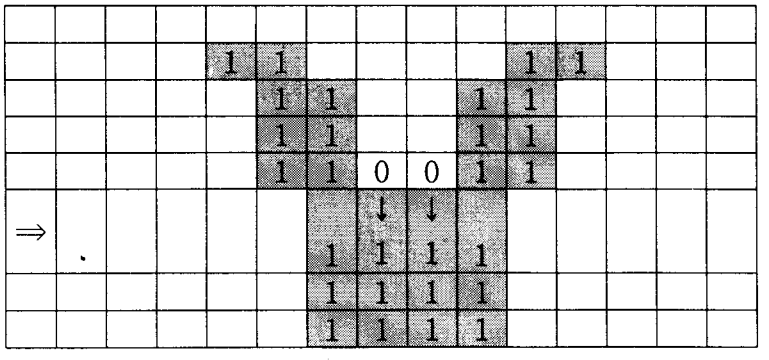
⇒ : 찾고자 하는 점의 위치 (끝점)

그림 2.15 Segment Type 끝점 조사

(8) int check_branch(int, int)

① Y 자형 분기에 의해 새로운 세그먼트의 시작인지를 검사

② 새로운 세그먼트의 시작이면 new_seg()를 호출함



⇒ : 찾고자 하는 점의 위치 (시작점)

그림 2.16 Segment Type 시작점 조사

(9) int *new_seg(int)

① 구조체 변수에 세그먼트의 시작점에 해당하는 X, Y 좌표를 기록함

② 최초 수행이면 calloc()를 이용하여 세그먼트 순서를 기록하기 위한 메모리를 할당

② 최초 수행이 아니면 realloc()을 이용하여 기록장소의 크기를 적절히 조절

④ 기록 장소를 항상 정렬된 상태로 유지 (sort() 호출)

seg[0]	seg[1]	seg[2]	. . . . .	seg[49]
sx	sx	sx	. . . . .	sx
sy	sy	sy	. . . . .	sy
ex	ex	ex	. . . . .	ex
ey	ey	ey	. . . . .	ey
type	type	type	. . . . .	type
cutpointx	cutpointx	cutpointx	. . . . .	cutpointx
cutpointy	cutpointy	cutpointy	. . . . .	cutpointy

그림 2.17 struct segment seg[SEG_SIZE]

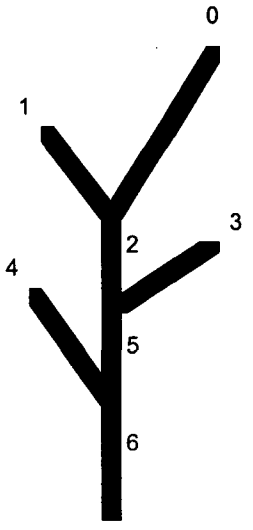


그림 2.18 segment와 seg_id

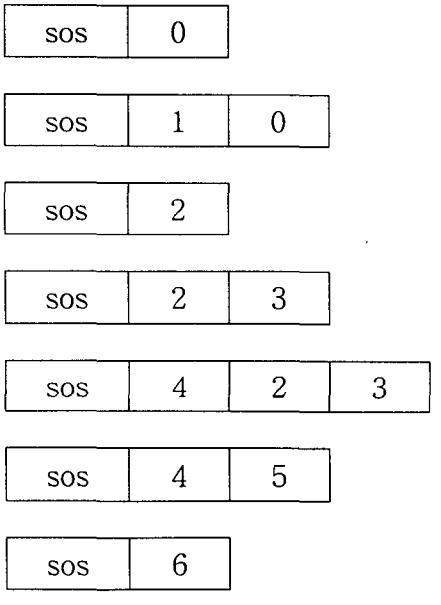


그림 2.19 sos 의 상태 변화

(10) int *end_seg(int)

① 세그먼트의 끝이 발견된 경우 세그먼트의 순서를 유지하는 배열에서 그 세그먼트의 id를 제거

(11) int *sort(int *, int)

① 배열을 X 좌표 값으로 정렬함 (Quick Sort)

(12) void skip_blank(void)

① '1'을 발견할 때까지 X 좌표 증가시킴

(13) void print_line(char *)

① 파일로 저장된 결과를 화면상에 출력

5. Branch Finder Program의 상수 & 변수

1) Predefined Constants

```
#define   FOUND_START_TIP   10
#define   NO_START_TIP       0
#define   FOUND_END_TIP     20
#define   NO_END_TIP         0

#define   YES                1
#define   NO                 0
#define   TRUE               1
#define   FALSE              0
```

```
#define    WIDTH          512        /* size of image  */
#define    HEIGHT         480
#define  SEG_SIZE         50
     /* available maximum size for segment number */
#define    PCP_SIZE       50
```

2) Variables

```
struct segment   {
      int sx;      /* 한 세그먼트의 시작점 x 좌표 */
      int sy;      /* 한 세그먼트의 시작점 y 좌표 */
      int ex;      /* 한 세그먼트의 끝점 x 좌표 */
      int ey;      /* 한 세그먼트의 끝점 y 좌표 */
      int type;    /* 세그먼트가 잎인지 줄기인지 구별 */
                   /* type == 200 이면 잎 */
                   /* type == 100 이면 줄기 */
      int cutpointx;    /* 줄기 cuttiing point 의 x 좌표 */
      int cutpointy;    /* 줄기 cuttiing point 의 y 좌표 */
   };


extern   struct   segment  seg[SEG_SIZE];
extern   int      x,y;
extern   int      seg_size ;
extern   int      seg_id;
extern   int      seg_ptr;
extern   int      start;
extern   int      pcp;
    /*  check point of the previous line  */
```

```
extern    int     ncp_flag;
    /* flag to indicate that the next line is a new segment  */
extern    int       remain_new_flag;
extern    int       remain_end_flag;
extern    int       pcp_flag;
extern    int       remain_seg;
extern    int       *sos, *temp1, *temp2, *temp3;
extern    FILE      *out;
extern    int       maxsize;
extern    int       cnt_w, cnt_z, chip_flag, chip_size;
```

제 3 절 조직배양실 소식물체 인식 프로그램

1. 시스템 개요

소식물체 인식용 화상처리 시스템에 대해 전반적인 설명을 한다.

① Window95 환경에서 PC를 Booting을 한다.
② Visual Studio Icon을 Click하여 Microsoft Developer Studio를 실행한다.
③ Menu Bar 상에서 "FILE" Pop Up Menu를 Click하면 Sub Menu가 나타난다. Sub Menu 중에서 "Open Workspace" Click하면 Open할 File을 선택하기 위한 File Dialog Box가 화면상에 출력이 된 후, 다음의 Directory 내의 File을 선택하여 소식물체 인식 응용 Program을 Loading 한다.

C:₩소식물체₩vt52₩vt52.dsw

④ Loading이 완료되면 Main Menu Bar 상에서 "Build" Pop Up Menu를 Click 하면 Sub Menu가 나타나고, Sub Menu 중에서 "Execute vt52.exe"를 Click하면 소식물체 인식 Program을 실행하게 되어서 다음과 같은 화면이 출력하게 된다.

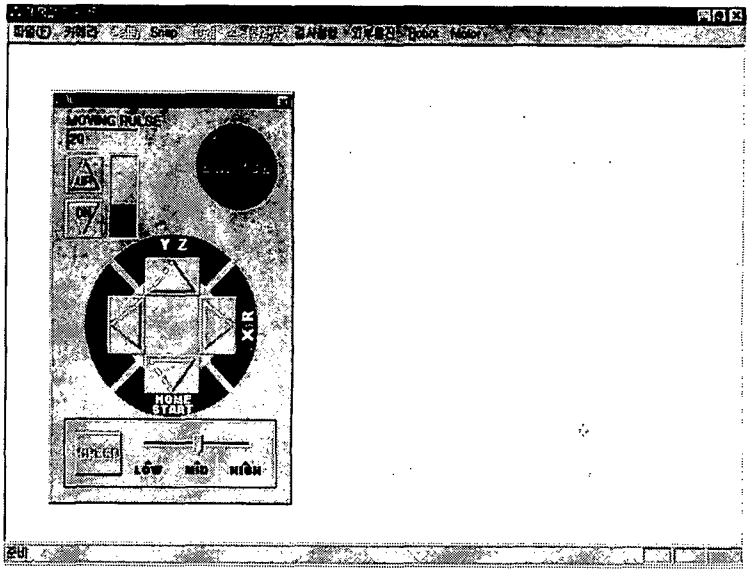


그림 2.20 소식물체 검사 프로그램 시작 메뉴

2. 소식물체 인식 S/W 사용법

1) 파일(F)

"파일" 메뉴를 선택하면 다음과 같이 화면상에 출력된다.

파일(F)
종료(X)

소식물 인식 응용 Program을 종료하기 위하여 Main Menu Bar 상의 "파일" Menu를 Click한 후 Sub Menu인 "종료"를 선택하면 소식물 응용 프로그램을 종료한다.

2) 카메라

(1) 모드 메뉴 : 사용할 카메라를 선택하기 위하여 사용하는 메뉴로서 다음과 같은 4 종류 카메라를 지원한다.

소식물 응용 프로그램을 실행한 후에 가장먼저 사용할 카메라를 선택해야 한다.

카메라	
모드 ▶	RGB
Cam No ▶	PAL
	RS170
	CCIR

(2) Cam No 메뉴 : Board에서 최대 4대의 카메라가 접속할 수 있어 화상을 입력할 Camera를 선택하는 메뉴이다. Default Channel로서 첫번째(One)가 선택된다.

카메라	
모드 ▶	
Cam No ▶	One
	Second
	Third
	Four

3) Calibration

Camera의 좌표계와 Robot 좌표계와의 상관관계를 설정하는 기능을 수행한다. 즉, 카메라 좌표계인 화소와 Robot 좌표계인 mm 단위의 상관

관계, Scale Factor를 구하는 단계이다. 또한, Robot이 소식물을 절단하는 위치와 Camera가 소식물체를 인식하는 위치가 달라서 각 좌표계의 절대점을 Link 시킬 수 있는 Calibration Tool이 필요로 하게 된다. Calibration Tool에는 각 좌표계의 원점을 설정할 수 있도록 가운데에 십자선이 그려져 있다. Main Menu Bar에서 "Calib Menu"를 Click하면 다음과 같은 Dialog Box가 화면상에 표시된다.

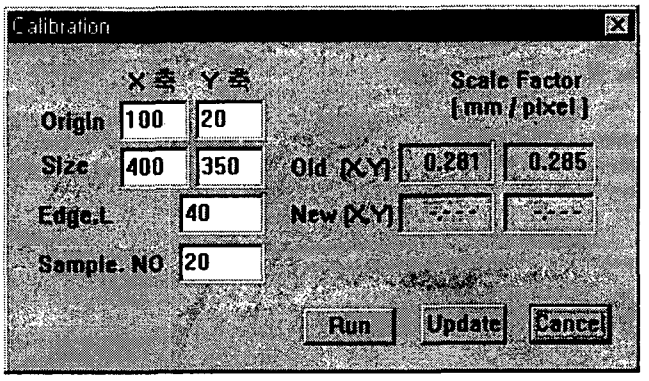


그림 2. 21 Calibration 메뉴

(1) Origin(X,Y) : 화상 좌표계에서 좌측 상단 Corner의 위치를 나타내는 좌표로서 Calibration Tool을 포함할 수 있도록 위치값을 입력시킨다.

(2) Size (X,Y) : Origin(X,Y)에서부터 검사할 영역의 크기를 지정하기 위한 입력변수로서, X Size는 화면상에서 우측 방향으로 값이 증가하고, Y Size 는 화면상에서 아래 방향이 증가하는 방향이다. 또한 원점으로부터 X와 Y의 크기의 입력에는 다음과 같은 크기의 제한이 있다.

$$0 < X\ Org + X\ Size < 640$$
$$0 < Y\ Org + Y\ Size < 480$$

이 때 원점의 위치와 크기를 입력할 때에 화면상에는 Rectangle이 그려진다. 이 사각형의 크기를 보면서 적절한 값을 입력시킨다.

(3) Edge.L : Calibration Tool 화상에서 Edge들을 추출하기 위한 기준값으로서 입력된 값보다 크면 Edge로서 간주된다. 입력범위는 0에서 255까지 가능하나 통상적으로 8 정도의 고정적인 값을 입력시킨다.

(4) Sample No. : R01 내에서 추출할 Edge들의 간격을 지정하는 변수로서, R01의 원점(X,Y)이 기준위치이고, 이 위치로부터 입력된 Sample 값을 더한 위치에서 Edge들을 추출한다. 이 값은 통상적으로 5로 고정한다.

(5) Old(X,Y) : 1 화소에 대한 절대적인 거리를 나타내는 단위로서, 기본단위는 mm 이다. 이 값은 구해진 기존의 값으로서 만일 Camera와 Lens를 변경하거나 재설정이 필요한 경우 이 값은 의미가 없다.

(6) New(X,Y) : 이 값은 Calibration Dialog Box 상의 "Run" Button을 눌러서 Calibration을 실행한 결과을 표시한하는 새로운 값이다. 새로 구해진 이 값을 앞으로 계속 사용하기 위해서는 "Update" Button을 사용한 경우에만 System에 저장되어서 사용이 가능하나, "Cancel"을 누른 경우에는 새로이 구해진 값은 사용되지가 않고, 기존의 Old(X,Y)상의 값을 사용한다.

(7) Run : (1) ~ (4) 번 까지의 입력을 완료한 후에 "Run" Button을 누르면, Calibration을 실행하고, (6)번의 New(X,Y) 값을 Update하여 표시한다. 또한 화면상에 찾아진 Calibration Tool의 Tool의 중심을 표시하여 Camera의 원점을 표시하고, 찾아진 Edge들을 화면상에 표시한다.

(8) Update : Calibration 작업을 완료한 후에 새로이 구해진 값들을 앞으로 반영할 경우에 "Update" Button을 눌러서 Calibration Dialog를

Exit한다.

(9) Cancel : Calibration 작업을 완료한 후에 새로 구해진 값들을 앞으로 반영하지 않기를 원하면, "Cancel" Button을 눌러서 Calibration Dialog를 Exit 한다.

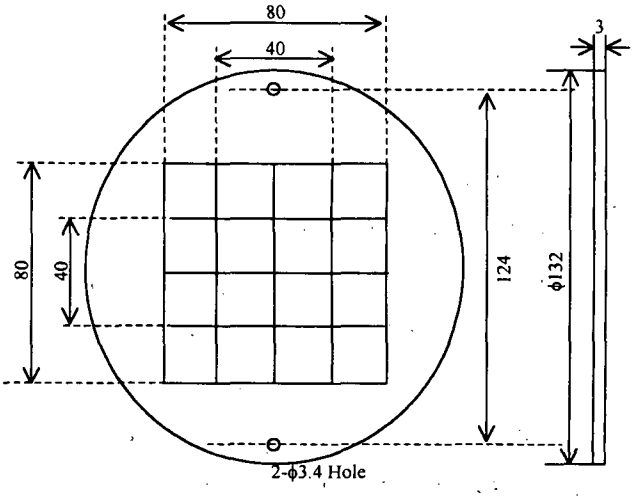


그림 2.22 비전 캘리브레이션 플레이트

4) Snap

Off-line의 화상처리를 위해 처리할 화상을 On-Board 상의 Frame Buffer에 저장하거나, Frame Buffer의 화상을 Hard Disk에 저장을 하여 보관하는 기능을 수행한다.

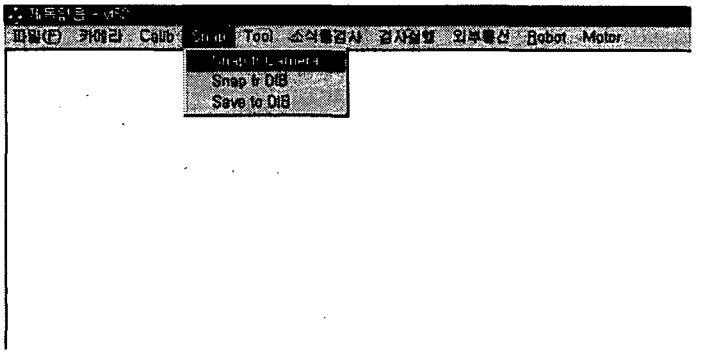


그림 2.23 카메라 Snap 메뉴

(1) Snap fr Camera : Camera로부터 Live 화상이 Frame Buffer에 저장된다. 저장된 화상은 Display Console을 통하여 화상 처리된 Graphic Date와 함께 화면상에 출력한다. Display Console은 최대 4개까지 설정되어 있고, 5 번째 Frame은 다시 첫 번째 Console에 출력된다.

(2) Snap fr DIB : HDD에 저장된 화상(BMP Format)을 Frame Buffer에 Loading한다. Loading할 화상 File은 File Dialog Box를 통하여 선택하고, Display Console을 통하여 화면상에 출력된다.

(3) Save to DIB : Frame Buffer의 화상은 향후의 화상처리를 위하여 저장할 필요가 발생하는데 이 때 저장할 File 명은 File Dialog Box를 통하여 선택되고, Frame Buffer상의 화상은 HDD에 BMP Format으로 File 명으로 저장된다. 이 때 한 화상의 Size는 256K Byte 정도 소요된다.

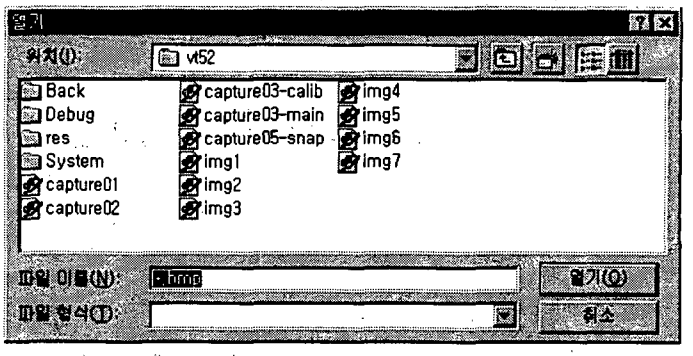


그림 2.24 이미지 파일 열기 메뉴

5) TOOL

"TOOL" 메뉴는 일반적으로 화상의 특징을 추출하기 전에 필요한 전처리 (Pre-Process) 기능으로서, 현재는 Cognex Tool을 사용하여서, Histogram과X/Y Projection 기능만을 구현하였고, 추후 확장이 필요하다. Tool 상의 Sub Menu의 기능들은 소식물의 인식에 사용되지 않는다.

(1) Histogram Tool Dialog Box

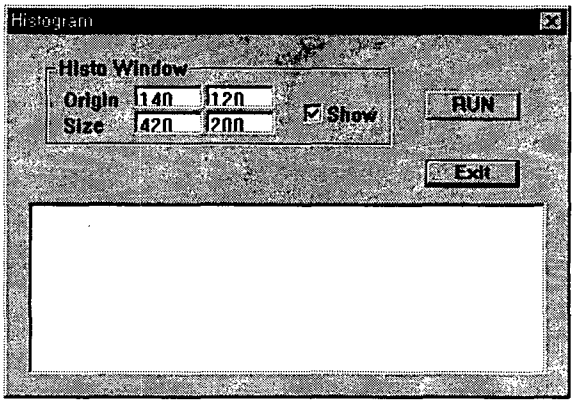


그림 2.25 Histogram Tool Dialog Box

(2) Projection Tool Dialog Box

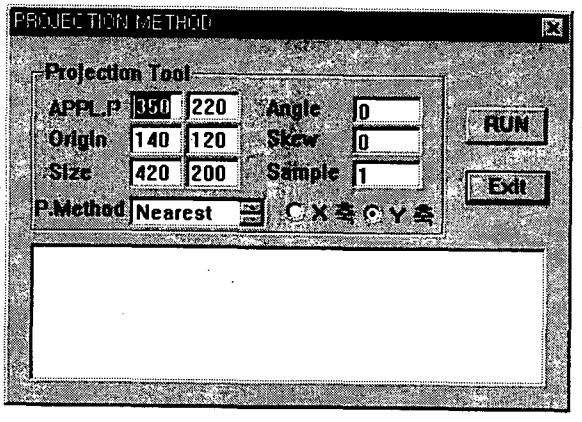


그림 2.26 Projection Tool Dialog Box

6) 소식물 검사

소식물의 Cutting 위치를 산출하기 위하여 화상처리를 수행하는 기능을 수행한다. Off-line을 통하여 인식 Parameter를 설정하는 Manual 작업과 설정한 Parameter를 가지고 검사를 수행하는 자동작업으로 구성된다.

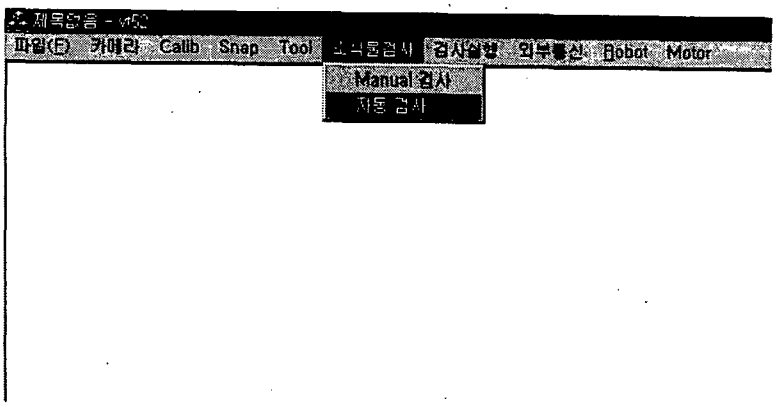


그림 2.27 소식물체 검사 메뉴

(1) Manual 검사 : 소식물의 최적 인식을 위하여 화상처리를 위한 변수들을 설정하고 그 설정된 값을 기준으로 화상처리를 단계별로 수행하여 인식을 위한 변수들의 설정이 올바로 되었는지를 판단할 수 있다. "Manual 검사" Button을 Click하면 다음과 같은 소식물 인식 변수 설정을 위한 Dialog Box가 화면상에 표시된다.

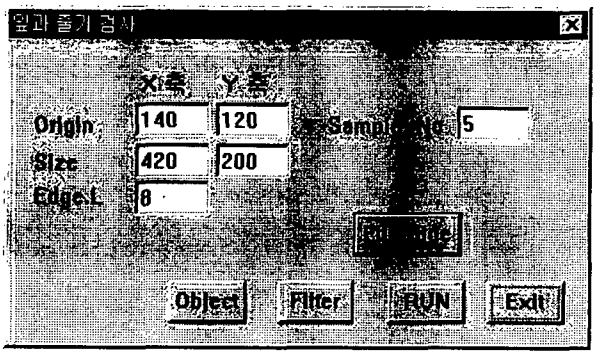


그림 2.28 잎과 줄기 검사 메뉴

① Origin(X,Y) : 화상 좌표계에서 좌측 상단 Corner의 위치를 나타내는 좌표로서, 검사영역이 소식물을 포함할 수 있도록 위치값을 입력시킨다.

② Size(X,Y) : Origin(X,Y)에서부터 검사할 영역의 크기를 지정하기 위한 입력 변수로서, X Size는 화면상에서 우측 방향으로 값이 증가하고, Y Size는 화면상에서 아래 방향이 증가하는 방향이다. 또한 원점으로부터 X와 Y의 크기의 입력에는 다음과 같은 크기의 제한이 있다.

$$0 < X\ Org + X\ Size < 640$$
$$0 < Y\ Org + Y\ Size < 480$$

이 때, 원점의 위치와 크기를 입력할 때에 화면상에는 ROI Rectangle이 그려진다. 이 사각형의 크기를 보면서 적절한 값을 입력시킨다.

③ Edge.L : 화상에서 소식물의 Edge들을 추출하기 위한 기준값으로서 입력된 값보다 크면 Edge로서 간주된다. 입력 범위는 0에서 255까지 가능하나 통상적으로 8 정도의 고정적인 값을 입력시킨다.

④ Sample No. : R01 내에서 추출할 Edge들의 간격을 지정하는 변수로서, R01의 원점(X,Y)이 기준위치이고, 이 위치로부터 입력된 Sample 값을 더한 위치에서 Edge들을 추출한다. 이 값은 통상적으로 5로 고정하여 사용한다.

⑤ Run : 앞 단계까지 입력을 완료한 후에 "Run" Button을 누르면, Edge Level로 설정된 값을 기준으로 R01이 영역 내에서 Edge들을 찾는다. 다음과 같은 수식을 기준으로 경계점들을 추출하여 화면상에 Edge들을 표시한다. abs(P(i,j+1) - P(i,j-1)) > Edge Level을 만족하면 P(i,j)는 Edge로서 간주되고, P(i,j+1)가 P(i,j)보다 작으면 파란색으로 화면상에 표시되고, 반대로 P(i,j-1)는 R01 내에서 (i,j-1)번째 화소에서의 Gray Level이고, P(i,j)는 (i,j)번째의 화소에서의 Gray Level, P(i,j+1)은 (i,j+1)번째의 화소에서의 Gray Level 값이다.

⑥ Filter : Run 수행의 결과로 찾아진 Edge들 중에서 중복되는 Edge들을 제거하는 과정으로서, 중복된 Edge는 Virtual Edge로서 추후 간주하고, 노란색으로서 화면상에 표시한다.

⑦ Object : 앞 단계에서 추출된 Real Edge를 갖고서 소식물 줄기나 잎의 한 Segment Object들을 만들어내고, 이웃하는 Segment Object들 간의 Chain Link를 만들어서 줄기와 잎의 교점을 찾아낸 후, 각 교점간의 연결 Link의 중심을 Cutting Point로서 간주하여 화면상에 Orange 색으로서 표시한다.

⑧ Exit : 소식물 인식 작업을 완료한 후에 새로 설정한 인식변수들을 소식물 인식에 반영하기 위한 System Data로서 등록을 하고 소식물 인식 Dialog Box를 Exit 한다.

(2) 자동검사 : Manual 검사 Mode에서 설정된 값을 기준으로 Camera로부터 소식물의 화상을 입력받아서 소식물의 Edge 추출에서

Cutting의 산출 및 표시 까지를 자동으로 수행한다.

7) 검사실행

"검사실행" 메뉴는 RS232C(COM2) Port를 통하여 Robot Controller로부터 명령을 받아서, 명령에 대하여 Action을 수행하고, 수행된 결과를 다시 Robot Controller에 전달하는 기능을 수행한다. Robot과의 통신 Protocol은 별첨을 참조 바랍니다. "검사실행" 메뉴는 다음과 같은 조건에 의하여 선택이 가능하게 된다. 즉, Gray 색상의 메뉴에서 선택 가능한 메뉴로 Enable 된다. "검사실행" 메뉴는 Camera의 모드 Sub Menu에서 Camera Type을 선택하고, 외부통신 메뉴를 선택해야만 Enable된다.

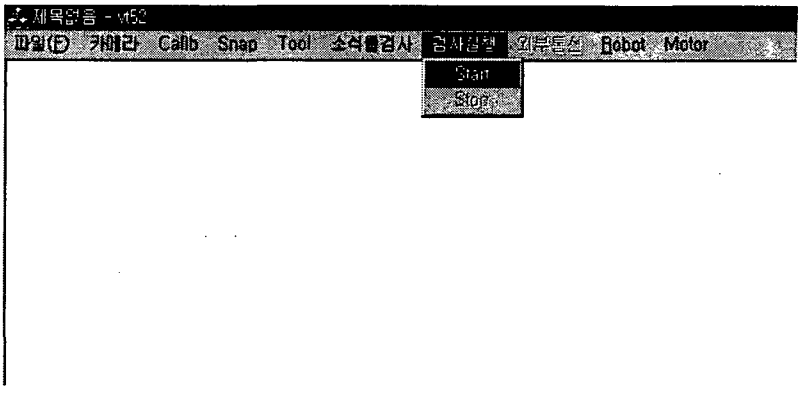


그림 2.28 검사 실행 메뉴

■ Start/Stop : Vision System이 Off-line에서 On-line 모드로 전환을 하게 되고 검사실행의 "Stop" Sub Menu만이 선택이 가능하고, 그 이외의 모든 메뉴는 선택이 불가능하다. On-line 모드에서는 Robot으로부터 전달되는 명령에 대응하는 기능만을 수행한다. Off-line Mode로 전환을 위하여서는 "Stop" Sub Menu에 의하여 On-line Mode를 종료한다. 이 때 "외부통신" 메뉴를 제외한 모든 메뉴가 선택이 가능하다.

8) 외부통신

Robot Controller와의 통신을 위하여, 통신 Port(COM2)를 초기화하고, Serial Port로부터의 송수신을 담당하고, 수신 명령을 해석하고, 명령을 수행하는 Command Interpreter Thread를 한다. "외부 통신" 메뉴는 한번만 선택이 가능하도록 선택한 후에 Gray 색상으로 변한다.

제 3 장 조직배양실 소식물체 핸들링 시스템

제 1 절 소식물체 절단작업에 레이저 이용

조직 배양실 증식작업에서 사용할 수 있는 절단 메카니즘은 청정 환경을 유지할 수 있어야 하고 작업 중에 혹은 작업 후에 세정작업에 편리한 구조를 가져야 한다. 그러나 레이저를 사용하면 커터에 대한 세정 공정이 필요 없게 된다. 실제로 CO2 레이저를 사용하여 현장적용 가능성에 대해 실험을 하였다. 이 실험에 이용한 레이저 절단 실험장치는 그림 3.1과 같다.

- CO2 Laser Cutter
  - 출력 : 70 ~ 90 Watts, 연속발진(CW)
  - 속도 : 0.5 m/min
  - 보조가스 : 공기
  - CO2 laser : 파장 10.6 μm
  - Robin-Sinar(RS1500)

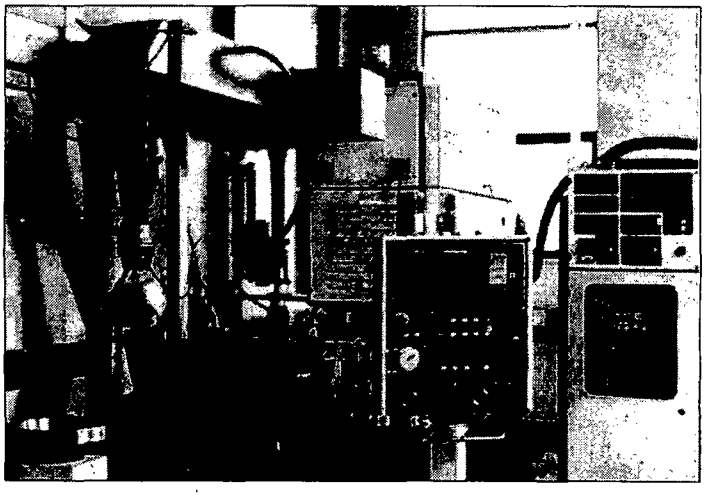


그림 3.1 CO2 레이저 장치

실험결과에 의하면 씨감자 줄기를 절단하기 위해서는 CO2 레이저 80 Watt의 파워가 필요하였고, 씨감자 줄기를 절단한 단면을 20 배 확대하여 본 것이 그림 3.2와 같다. 그림 3.3에서 보는 바와 같이 절단 단면이 비교적 깨끗하게 잘 절단된 형태로 나타났다. 간혹 단면 부위에서 약간 탄 부위가 있으나 이것은 식물의 생존이나 성장에 영향을 미치는 것이 아니고 일종의 소독 성질을 갖는다.

그림 3.2 CO2 레이저를 이용한 절단 단면

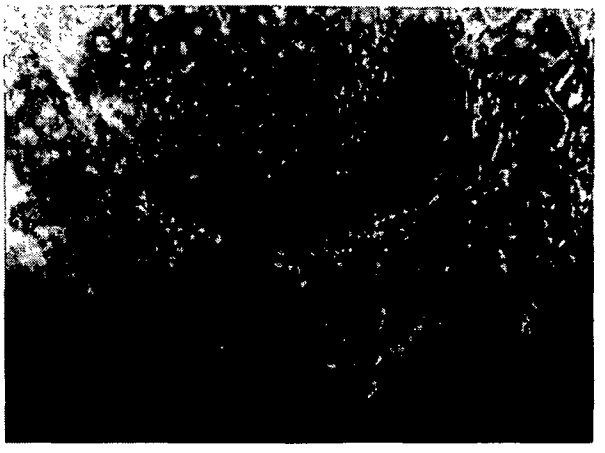


그림 3.3 레이저 절단 단면 80배 확대

레이저 절단은 절단면이 깨끗한 형태로 잘 절단이 되었고 사용하기에 기구적으로 편리한 점도 있으나, 실제 레이저 전체 장치가 크고 장비 가격이 비싸다는 단점을 가지고 있다. 씨감자 중식작업의 절단 공정에 레이저를 사용하기에는 많은 비용이 들기 때문에 실제 적용하기에는 어렵다. 레이저를 사용하면 커터에 대한 세정작업이 필요 없지만, 그리퍼에 대한 세정작업은 여전히 남아 있게 마련이다. 그러므로 그리퍼에 대한 세정작업과 커터에 대한 세정작업을 동시에 할 수 있는 구조로 하면 세정작업의 중복성은 피할 수 있다.

제 2 절 파지 및 절단 메카니즘

앞서 설명한 바와 같이 씨감자 줄기 절단 작업에 레이저를 사용하기에 비용이 너무 많이 소요되고, 장비 자체가 너무 크기 때문에 커터와 그리퍼의 세정작업을 동시에 할 수 있는 구조를 갖는 기계식 파지 및 절단 메카니즘에 대해 연구를 하였다

그림 3.4와 같이 로봇의 앤드-이펙트에 공압 그리퍼를 장착하여 여기에 소형의 커터와 그리퍼를 부착하여 공압을 사용하여 커터와 그리퍼가 동작하도록 하였다. 이 커터와 그리퍼가 좁은 공간에서 소식물체를 파지하여 절단하고 핸들링 할 수 있도록 하여야 하고, 공압 그리퍼에 부착할 수 있도록 콤팩트하게 커터와 그리퍼의 설계가 필요하다. 현재 RV-E2 로봇의 앤드-이펙트 구조상 커터와 그리퍼를 분리할 경우 공압 그리퍼에 부착할 수 있는 구조를 가지고 있지 못하다. 즉, 현재의 앤드-이펙트 구조상 커터와 그리퍼 2개를 동시에 부착하여 동작시킬 수 있는 구조가 아니다. 그리고 커터와 그리퍼를 각 각 따로 작동하도록 하는 경우, 공압 그리퍼도

2 개가 존재해야 하고, 이를 위한 Tool Changer도 필요하게 되므로 시스템이 복잡하게 된다. 그러므로 1 개의 공압 그리퍼에 커터와 그리퍼를 장착하여야 하기 때문에 커터와 그리퍼가 일체화가 될 수 있는 구조로 설계하였다. 커터는 소식물체가 절단시 절단 부위 이외에는 손상을 주는 곳이 없도록 설계하여야 하고, 또한 화염 또는 약품에 의한 세정 작업을 쉽게 할 수 있는 구조가 되도록 하여야 한다. 그러므로 절단기로는 의료용 가위를 이용하여 구성하였고, 그리퍼도 의료용 Forcep을 사용하였다. 커터와 그리퍼는 동일한 공압 그리퍼에 장착되어 하나의 공압 구동원을 사용할 수 있도록 하였고, 설계시 일체화 할 수 있도록 동일한 크기의 것을 선택하였다. 그리고 의료용 가위와 포셉은 가능한 소형 사이즈를 이용하여 좁은 공간에서 파지 및 절단작업을 용이하게 할 수 있도록 하였다. 그림 3.5은 설계된 커터와 그리퍼의 구조를 나타낸 것이다.

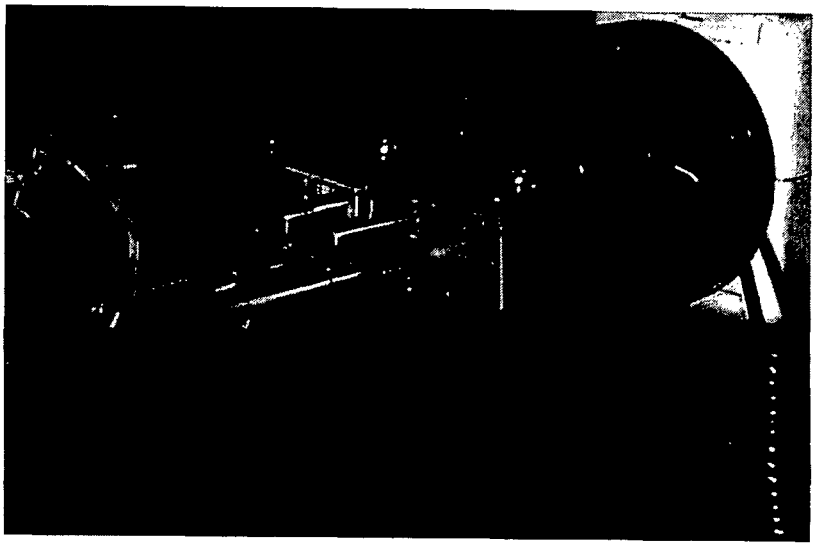


그림 3.4 그리퍼 및 커터가 소식물체를 잡고 절단하는 광경

그림 3.5에서 커터와 그리퍼는 평행하게 겹쳐 놓여 있는 모습이다. 연결되어 있는 노브 블록(A)의 내부는 그림 3.6과 같이 매우 복잡한 메카

니즘을 가지고 있다. 복잡한 메카니즘을 갖는 이유는 두 개의 노브 블록이 마주 닿을 때, 즉, 공압 그리퍼가 닫히게 될 때 커터와 그리퍼도 같이 닫히게 된다. 이 사이에 소식물체의 줄기가 들어가서 커터로는 절단이 되고 그리퍼에 남은 줄기 부분을 잡게 된다. 여기서, 고려해야 할 것은 소식물체의 줄기가 그리퍼의 파지력에 의해 손상되지 말아야 한다. 따라서, 노브 블록 내에 B와 같이 소프트한 스프링이 연결된 슬라이딩 구조가 여유를 갖도록 하여 줄기가 상하지 않도록 적절한 연약 강도를 갖는 스프링을 사용하였다. 노브 블록 A와 B에 대한 자세한 구조는 그림 3.6와 같다

그림 3.6에서 A는 절단기 및 그리퍼가 연결되는 노브 블록이며, B는 그리퍼가 여유있게 잡을 수 있도록 스프링과 연결되어 A내에서 움직이는 장치이다. 스프링은 소식물체가 상하지 않으면서 소식물체를 잡을 수 있도록 스프링 강도를 고려해서 설치하였다.

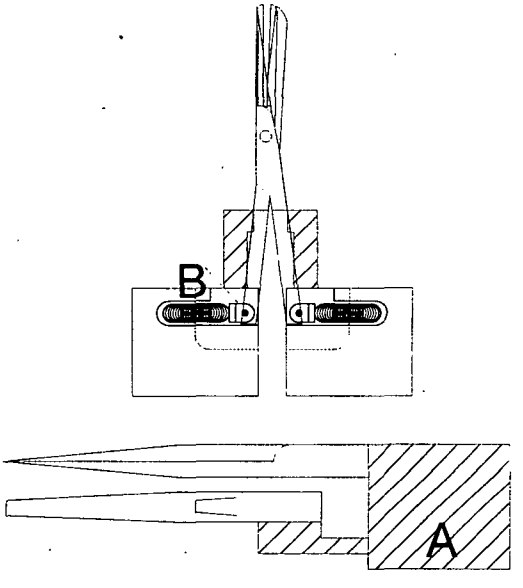


그림 3.5 커터와 그리퍼의 구조

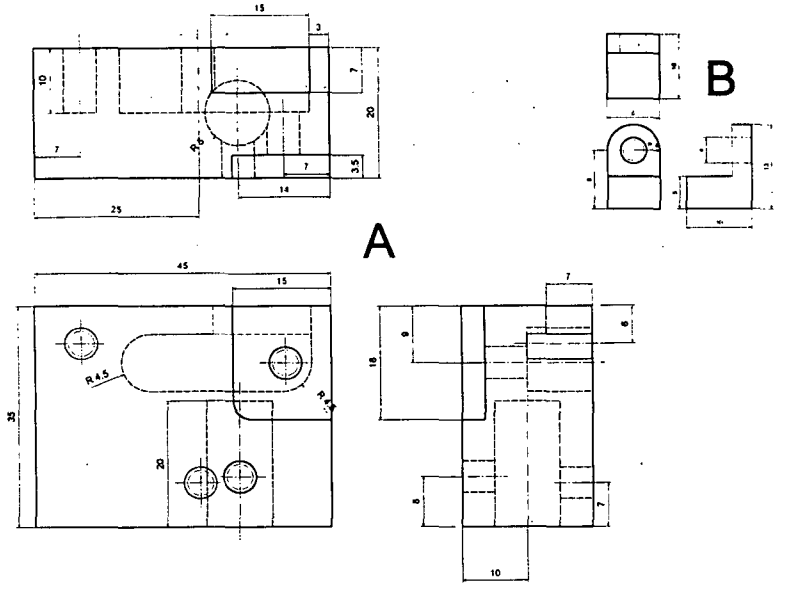


그림 3.6 노브 블록(A)와 B의 구조

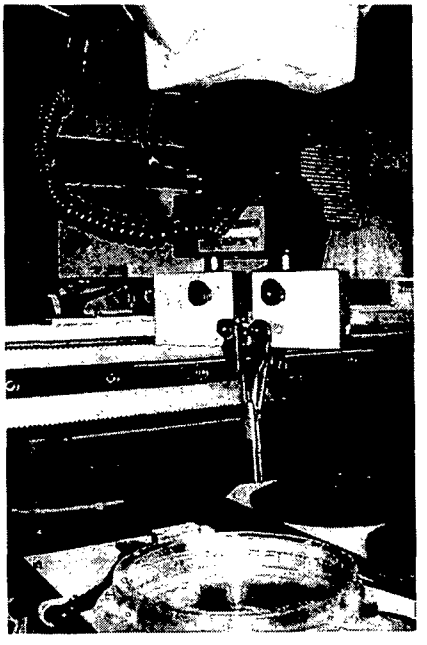


그림 3.7 파지 절단 그피퍼

제 3 절 조직배양실 핸들링 로봇 시스템

1. 시스템 구성

1) 전체 시스템 구성

조직배양실 씨감자 증식작업에 적용할 핸들링 시스템의 주요 구성 단위 시스템은 다음과 같다.

- Petri Dish 핸들링 XZ 로봇 시스템
- Petri Dish 핸들링 그리퍼 시스템
- Petri Dish Loading/Unloading 시스템
- 핸들링용 6축 다관절 로봇 시스템
- 화상처리 시스템

시스템 전체 구성 개념은 그림 3.8과 같고, 소식물체 조직 배양실에서 배양된 씨감자 페트리 디시(Petri Dish)가 이송되어 증식 작업실의 스테이션 1A에 도착할 때 이미 증식 작업실에서는 빈 페트리 디시가 스테이션 4A에 이송되어 와서 대기한다. 씨감자 페트리 디시가 증식 작업위치에 스테이션 1A에 도착하여 페트리 디시 홀딩장치 1C가 페트리 디시를 홀딩하면 XZ 로봇이 와서 페트리 디시 뚜껑을 잡고 회전하여 Open를 하여 뚜껑을 적재 스테이션 2A에 놓게 된다. 그 다음 XZ 로봇이 다시 와서 뚜껑이 Open된 페트리 디시를 작업 비전 스테이션의 백라이팅(Back Lighting) 상에 놓게 된다.

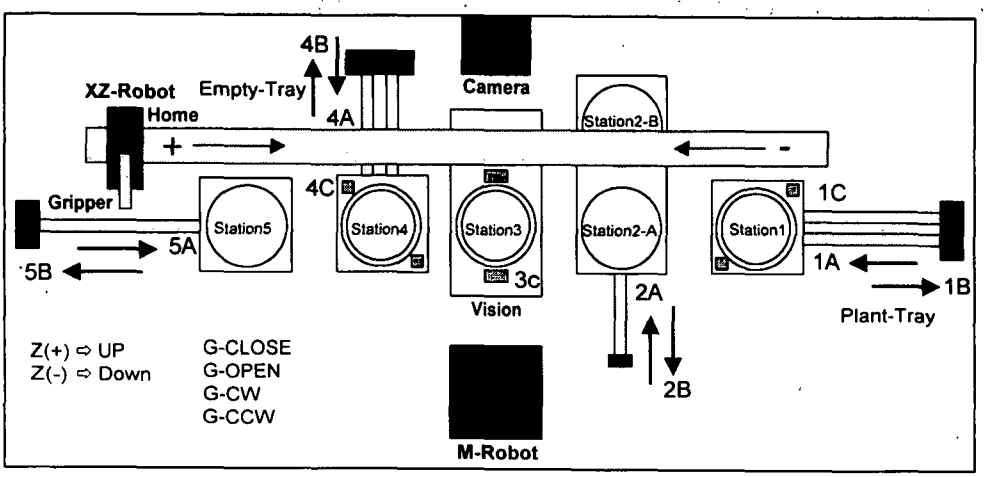


그림 3.8 로봇 핸들링 시스템 구성

페트리 디시 홀딩장치가 페트리 디시를 홀딩하고 소식물체 인식 시스템에 의해 씨감자의 절단위치를 확인하게 되면 핸들링 로봇이 이동하여 와서 파지 및 절단 동작을 완료한 후 절단된 소식물체 부위를 스테이션 4A에서 대기하고 있는 빈 페트리 디시에 투입(이식)하는 작업을 되풀이하게 된다. 파지, 절단 및 투입 작업이 완료되면 투입 완료된 페트리 디시는 배양실로 이송되게 된다. 그리고 백라이팅 상에 있는 페트리 디시에 대해 작업이 완료되면 이 페트리 디시를 적재 장소로 이동시키고 다음 페트리 디시에 대해 작업을 반복한다. 그림 3.9는 전체 핸들링 시스템이 구성된 사진을 보여 준다.

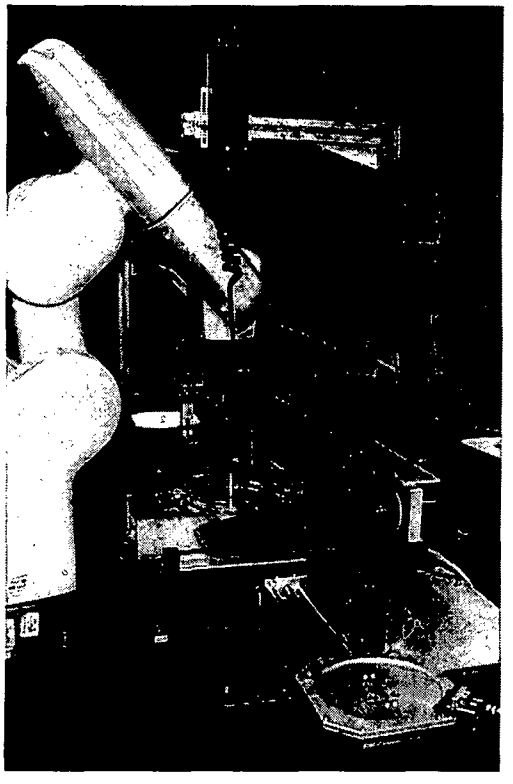


그림 3.9 소식물체 핸들링 시스템

2) Petri Dish 핸들링 그리퍼 및 XZ 로봇의 Z축 시스템

페트리 디시 핸들링 그리퍼는 그림 3.10과 같이 XZ 로봇의 Z축에 부착되어 있으며, Z축의 동작은 스텝 모터와 타이밍 벨트에 의해 상하운동을 한다. 페트리 디시 핸들링 그리퍼는 공압 구동에 의해 그리퍼 자체가 90도 회전(CW,CCW)을 할 수 있도록 되어 있으며, 또한 그리퍼 핑거의 Open, Close를 위한 직선 운동을 공압 구동에 의해 행해진다.

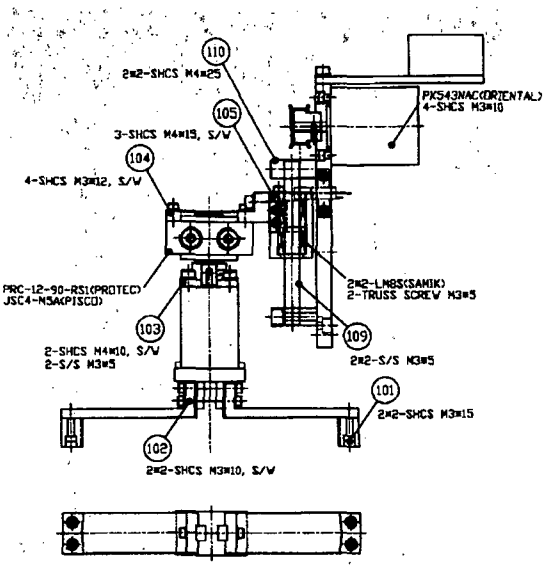


그림 3.10 Petri Dish 핸들링 그리퍼 및
XZ 로봇의 Z축 시스템

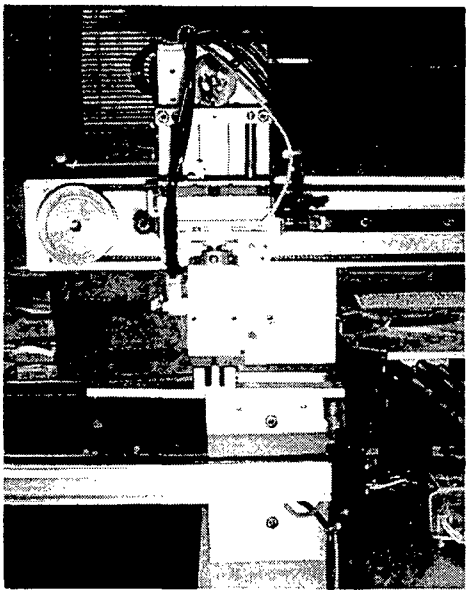


그림 3.11 페트리 디시 핸들링
그리퍼 및 Z축 시스템

3) Petri Dish 핸들링 XZ 로봇의 X축 시스템

페트리 디시를 각 스테이션 간에 이동시켜주기 위해 XZ 로봇의 X축이 동작하여 이동작업을 한다. 이때 페트리 디시 핸들링 그리퍼가 부착된 Z축도 함께 동작을 한다. X축의 구동은 LM 가이드, 스텝모터, 타이밍 벨트에 의해 구동한다.

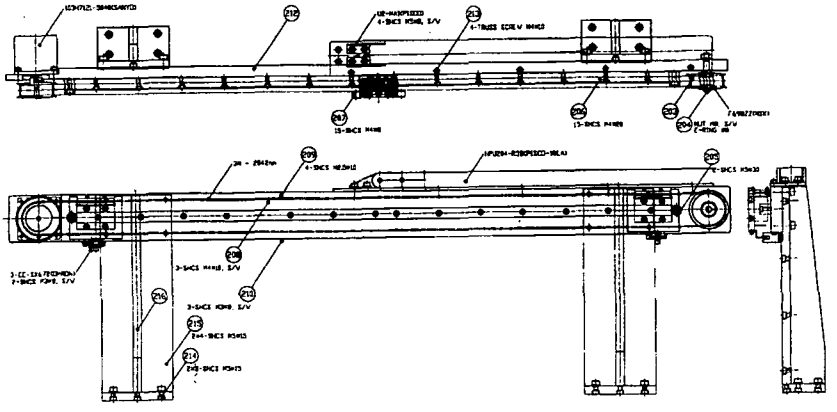


그림 3.12 Pteri Dish 핸들링 XZ 로봇의 X축 시스템

4) Petri Dish Loading/Unloading Transfer 시스템

그림 3.8, 3.13에서와 같이 5 대의 스테이션은 각각 하는 기능의 역할이 다르다. 각 스테이션은 공압에 의해 구동을 하며, 각 스테이션에는 다시를 홀딩할 수 있는 Jig 부분을 공압에 의해 구동되고 있다. 실제 현장에 적용할 때는 스테이션 1, 2, 4, 5에 대한 Loading/Unloading 장치를 부착하여 전체 시스템이 자동화, 무인화가 되도록 하여야 할 것이다. 스테이션 3은 비전 검사를 하는 스테이션이며, Back Lighting 조명장치가 하단에 부착되어 있으며 위쪽 방향에 지지대에 의한 카메라가 부착되어 있다. 여기서 소식물체 검사 과정이 이루어진다.

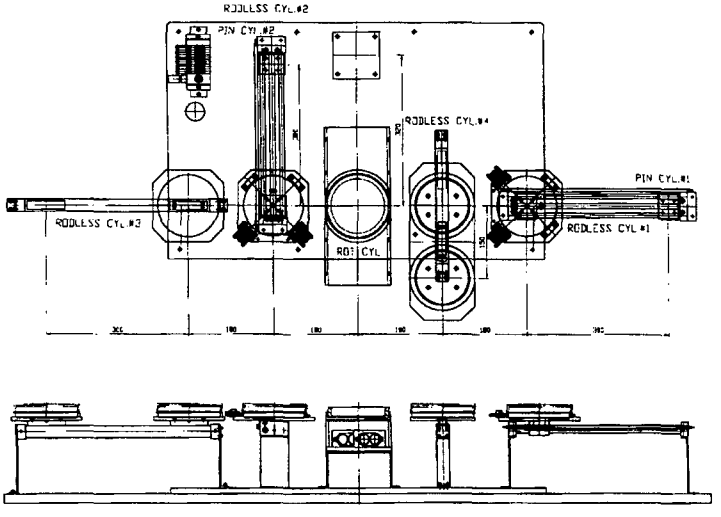


그림 3.13 Petri Dish Loading/Unloading Transfer System

5) 소식물체 핸들링 작업 공정 순서

소식물체 핸들링 작업을 위한 전체 시스템의 주요 구동순서는 그림 3.14와 같다.

Step1. Robot 준비동작 : XZ-Robot Home Position, M-Robot Home Position, Gripper Home Position
Step2. 작업 준비 : Plant-Tray loading(Station1,1B), Empty-Tray loading(4B, Covered Tray)
Step3. 작업시작 :
Station1 이동(1A 동작) ⇨ Fixture CLOSE(1C)
⇨ XZ-Robot 이동(Home ⇨ Station1, X=)
⇨ Gripper Z 방향으로 Down(카버 위치까지, X=)
⇨ Gripper-CLOSE ⇨ Gripper-CCW ⇨ Gripper-UP(Z=)
⇨ XZ-Robot 이동(Station2-A, X=)
⇨ Gripper-DOWN(카버 안착 위치, Z=) ⇨ Gripper-OPEN ⇨ Gripper-CW
⇨ Gripper-UP(Z=) ⇨ XZ-Robot 이동(Station1, X=)
⇨ Gripper-Down(Tray Grip 위치, Z=)⇨ Fixture Open(1C)
⇨ Gripper-CLOSE ⇨ Gripper-UP ⇨ Gripper 이동(Station3, X=)
⇨ Gripper-DOWN(Tray 안착위치, Z=) ⇨ Gripper-OPEN ⇨ Gripper-UP
⇨ Gripper 이동(Station4 위치, X=)
Step4. Vision-PROCESSING
Step5. Fixture-CLOSE(4C) ⇨ Gripper-DOWN(카버 Grip 위치, Z=)
⇨ Gripper-CLOSE ⇨ Gripper-CCW
⇨ Gripper-UP(Z=) ⇨ Gripper 이동(Station5, X=)
⇨ Gripper-DOWN(카버 안착위치, Z=)
⇨ Gripper-UP(Z=) ⇨ Gripper 이동(Home 위치, X=)
Step6. M-Robot 작업 시작 ⇨ M-Robot 반복작업(Station3 ⇔ Station4, 비전정보에 의한 Cutting and Planting)
Step7. M-Robot 작업 완료
Step8. Gripper 이동(Home ⇨ Station5, X=) ⇨ Gripper-DOWN(Z=)
⇨ Gripper-CLOSE ⇨ Gripper-UP(Z=)
⇨ Gripper 이동(Station4, X=) ⇨ Gripper-DOWN(Z=)
⇨ Gripper-CW(Cover Close) ⇨ Fixture-OPEN(4C)
⇨ Gripper-UP(Z=) ⇨ Gripper 이동(Station5, X=)
⇨ Gripper-DOWN(Z=) ⇨ Gripper-OPEN
⇨ Gripper-UP(Z=) ⇨ Gripper 이동(Home Position, X=)
⇨ Station5 이동(5B)
Step9. 1 사이클 작업 완료

그림 3.14 핸들링 로봇 시스템의 작업 동작순서

2. 소식물체 핸들링 로봇 시스템

일본 Mitsubishi사에서 제작한 RV-E2 로봇은 몸체가 작고 무게도 가벼워 중량이 작은 물체를 핸들링하기에 적합한 로봇이다. 로봇 제어기도 기존 로봇의 제어기보다 크기도 1/4 정도밖에 되지 않아 소형 물체 핸들링 작업에는 매우 적합한 환경을 갖추고 있다. 또한, 외부 기기와의 통신을 위해 RS-232C 포트를 갖추고 있어 PC등의 Host에서 로봇을 제어할 수 있다.

본 연구에서 사용하는 로봇 시스템은 RV-E2이란 로봇으로써, Teach Pendant를 통한 교시 및 작업수행 이외에도, PC를 Host로 하여 작업을 수행할 수 있다. 보통 교시는 Teach Pendant를 통해서 하고 작업수행은 Host에서 하는 것이 편리하다. 그 이유는 Teach Pendant만으로 프로그램 작업을 하는 방법이 Pendant 자체의 Data-Key 입력만으로는 너무 복잡하고, 다양한 작업을 지원하지 못하기 때문이다. 따라서, Host에서 RS-232C를 통하여 Data를 전송하는 방식의 Program을 구성해야 한다. PC에서 RS-232C를 통해 Data 전송하는 방법은 여러 가지가 있으나, Vision System과의 통합을 고려해 Windows98 환경에서 프로그램을 구성하였다.

본 연구에서는 화상처리를 위한 Vision System과의 원활한 연계를 위해서 Windows98 상에서 Host Program을 제작하였다.

1) Windows 98상에서의 RS-232C 통신

Windows98 상에서 RS-232C 포트를 통해 통신하는 프로그램을 구성하려면 다양한 Tool을 사용할 수 있다. 본 연구에서는 Visual C/C++

5.0을 이용하여 사용자가 간편하게 로봇 프로그램을 구성할 수 있도록 GUI(Graphic User Interface)를 이용한다. 즉, 모든 명령을 메뉴 방식으로 구성하여 사용자가 명령문을 외울 필요가 없게 하도록 한다. 이렇게 구성된 명령문은 즉시 RS-232포트를 통해 로봇제어기에 전달되어 로봇이 수행할 수 있도록 한다. 먼저, RS-232C를 통해 Data를 송수신하는 방법을 살펴보도록 한다. DOS상에서 이 기능을 구현하려면, BIOS를 이용해서 Interrupt를 사용하거나, 직렬포트 구동 H/W인 8255의 Register를 직접 조작해서 비동기 통신을 하여야 했으나, Windows에서는 이런 작업이 필요가 없다. 모든 외부장치는 하나의 File 형태로 존재하고, 이러한 장치는 OS Level에서 관리되기 때문에, 특정 프로그램이 외부장치를 직접 건드릴 수는 없고, File에 Data를 쓰는 것처럼 프로그래밍을 하면 OS가 나머지 일들을 해주기 때문이다. 즉, 특정장치를 사용하고자 한다면 해당 Device Driver만 제대로 설치하고, 프로그램 상에서는 해당 장치의 Handle만 넘겨주는 형식으로 사용할 수 있다. 다음은 RS-232C에 대한 Handle을 만들어주는 Routine이다.

```
ComDev = CreateFile(szDevice, fdwAccess, fdwShareMode,
                    lpsa, fdwCreate, fdwAttrsAndFlags, hTemplateFile);
```

상기의 CreateFile 문은 매우 복잡한데, 그 이유는 File을 만드는 데 뿐만 아니라 RS-232C 처럼 외부장치의 Handle도 만들 수 있는 범용성을 띠고 있기 때문이다. 상기의 문장으로 생성된 핸들을 사용하여 쉽게 해당 포트에 데이터를 쓸 수 있다. 그러나 외부장치, 즉 로봇 제어기에서 나오는 데이터를 읽어들이려면 여러 가지 방법이 존재한다. 비동기, 동기, Polling 방식, 그리고 Event-Driven 방식이 있을 수 있다. Polling방식은 계속 Port를 읽어봄으로써 Data를 읽는 방식으로 사용하기엔 쉬운 장점이 있으나 쓸데없이 실행시간을 잡아먹는 단점이 있다. 동기(Synchronous)방식은 한번 Data를 읽기 시작하면 Data가 들어올 때까지 계속 기다린다. 따라서, Data가 들어오는 시점과 그 Data를 읽는 시점이 정확하게 일치하므로 동기방식이라고 한다. 이런 방식은 DOS상에서 주로 구현하던 방식이

며 Multi-Tasking OS인 Windows에서는 특별한 경우를 제외하고는 잘 사용되지 않는 방법이다. 비동기(Asynchronous)방식은 Background에서 Data를 읽는 방식으로 Data 양이 많을 때 유용한 방식이다. 즉, 1000 byte를 읽고 다른 작업을 수행하고, 또 1000 byte를 읽는, Multitasking에 적합한 방법이다. 마지막으로 Event-Driven 방식이 있을 수 있다. 특정한 Signal이 Assert 되었을 때에만 Data를 읽는 방법이다. 이렇게 특정한 Signal이 Assert되었을 때 Data를 읽으려면 Masking을 하면 되는데 이때 사용하는 함수가 SetCommMask란 함수이다. 본 연구에서는 이러한 Event-Driven방식으로 Data를 읽는 방법을 채택했다.

2) RV-E2상에서의 RS-232C 통신

(1) 통신 사양

RS-232C 인터페이스를 사용할 경우에는 다음과 같이 PC측에서 통신을 설정해야 한다. 이 설정은 로봇제어기와 PC측이 같아야 하며, 동일 조건이 아닐 경우에는 정상적으로 통신을 할 수 없다. 표3.1은 통신사양이다.

항 목	내 용
Baud Rate	9600 bps
Data Bits	8 bits
Parity Check	Even Parity
Stop Bits	2 bits

표 3.1 RS-232C 통신 사양

(2) 각 신호선의 Timing Chart

RS-232C 인터페이스의 규격은 원래 전기적인 사양과 Connector 형상, 핀번호 등을 정한 것이다. 각 신호선의 사용방법과 통신 순서 등은 기계에 따라 각각 다르게 되어 있다. PC와 접속하는 경우에도 대응하는 신호선을 접속하는 것만으로는 정상적으로 동작하지 않을 수가 있으므로, PC에서 사용하는 신호선과 로봇제어기의 신호선의 기능을 잘 이해한 후 접속해야 한다. PC와 로봇제어기 사이의 데이터 송수신은 모두 ASCII Code로 실행된다.

(3) PC로부터 로봇제어기의 데이터 전송 타이밍

가. 로봇 제어기

ER(DTR), RS(RTS)를 모두 'H' Level로 하여 데이터를 기다린다. 종료코드가 입력되면 ER(DTR), RS(RTS)를 모두 'L' Level로 하여 수신한 명령어를 실행한다. 명령어 실행이 끝나면 다시 ER(DTR), RS(RTS)는 'H' Level이 된다. 종료코드는 "0x0D", 또는 "0x0D + 0x0A"이다.

나. PC측

DR(DSR)이 'H' Level일 때 데이터 전송을 실행한다. DR(DSR)이 'L' Level일 때 데이터를 전송하면 로봇제어기에서 알람이 발생한다.

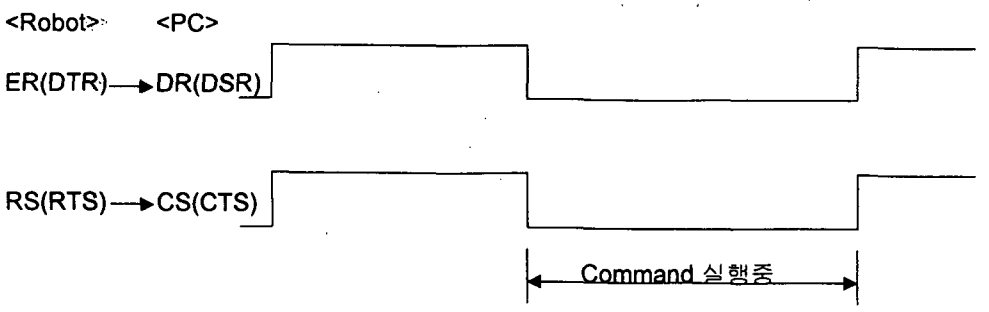


그림 3.15 데이터 전송 Timing Chart(PC→로봇 제어기)

(3) 로봇 제어기에서 PC로의 데이터 전송 타이밍

가. 로봇 제어기

ER(DTR)을 'H" Level로 한 후 데이터 전송을 개시한다. 종료코드를 전송한 다음에는 ER(DTR)을 'L' Level로 한다. 종료코드는 Parameter "CM0"로 설정할 수 있다.

나. PC측

ER(DTR), RS(RTS)를 모두 'H' Level로 하여 송신을 요구한 후, 로봇제어기 측으로부터 전송데이터를 기다린다.

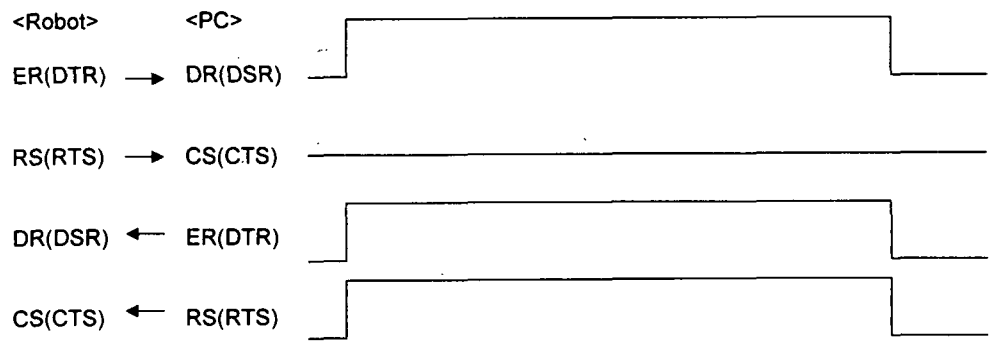


그림 3.16 데이터 전송 Timing Chart(로봇제어기→PC)

3) 핸들링 로봇 명령 구성

RV-E2 로봇의 제어기는 다양한 명령어를 갖추고 있는데, 이를 분류해 보면, 로봇을 움직이고 위치를 지정하는 등의 위치/동작 명령, 프로그램의 흐름을 제어하고 조건분기 등을 하는 프로그램 제어 명령, 로봇 핸드를 제어하는 핸드제어 명령, 외부의 다양한 기기를 제어하는 I/O제어 명령, RS-232C 포트를 읽는 RS-232C 읽기 명령, 그리고, 그 외의 명령 등이 있다. 또한 좌표계 설정, Workspace를 제어하는 등의 여러 가지 부수적인 설정을 할 수 있는 Parameter들이 있다. 이러한 명령과 파라미터들을 사용자가 쉽게 설정할 수 있으려면, 메뉴방식으로 사용하는 방법이 무척 편리하다. 예를 들어, 로봇을 위치#1에 위치시키게 하려면 우선 주메뉴에서 위치/동작 명령을 선택하고, 그 중에서 Move 명령을 선택한 다음, Move 명령에 필요한 Argument, 즉, 위치#1을 입력하면 1행의 입력이 끝나는 것이다. Windows98의 모든 메뉴는 이러한 방식으로 구성되어 있어서, 구성하기에 쉽다. 그림 3.17는 명령어 흐름의 계통도를 나타낸다.

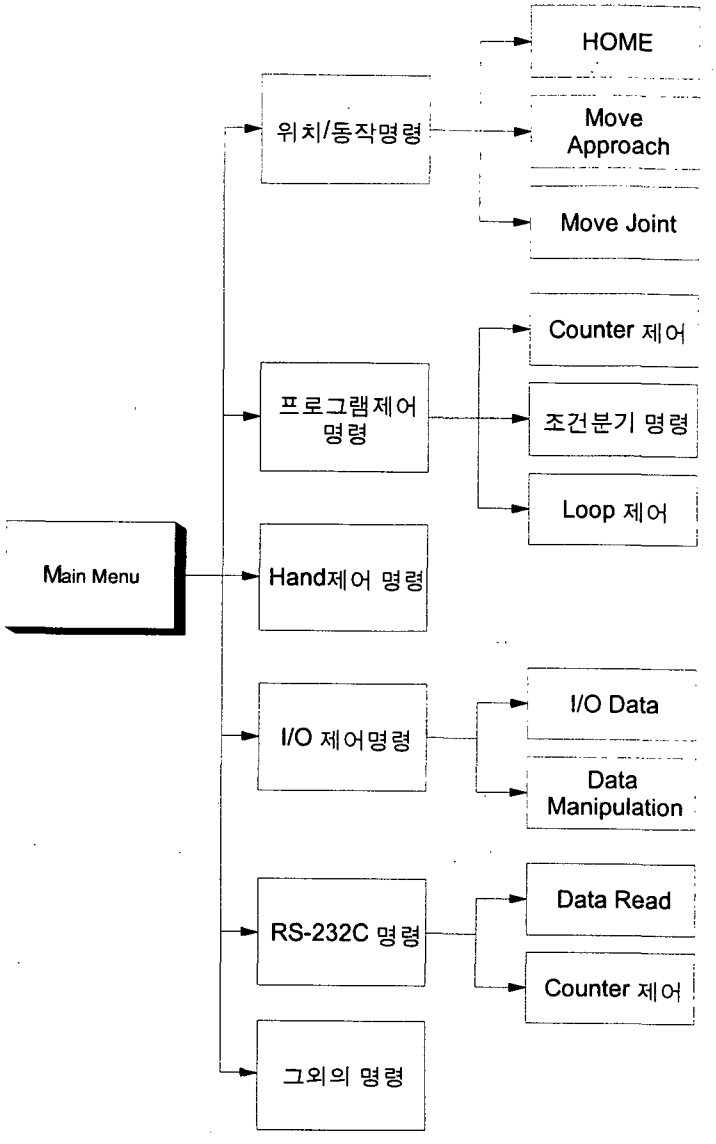


그림 3.17 명령어의 계통도

3) 로봇 호스트 프로그램

프로그램을 시작하면 그림 3.18과 같은 화면이 표시되고, 사용자는 먼저 명령어 종류에서 입력하고자 하는 명령어의 종류를 선택한다. 그러면 선택된 종류에 따라 명령문이 메뉴 박스에 나타난다. 필요한 명령문을 선택하면 그에 대한 설명문이 아래에 표시되어 설명서를 참조할 필요 없이 직접 프로그램을 구성할 수 있다. 인자에는 명령문마다 인자가 없을 수도 있고, 1 개 이상일 수도 있어서, 처음 프로그램을 할 때에는 사용자가 직접 입력을 해야한다. 그러나, 같은 인자가 사용될 때에는 그것을 기억하고 있어서, 다시 입력할 필요가 없다. 모든 프로그램의 구성이 끝나면, 전송버튼을 눌러 프로그램을 로봇제어기에 전송한다. 통신포트 버튼은 현재 사용하고 있는 Host Computer의 RS-232C Port에 대한 Setup을 하는 것이다. 현재 프로그램은 상술한 바와 같이 입력할 수 있으며, 또한 프로그램 Section에서 직접 Keyboard로 입력하는 것이 가능하다. 사용자의 입장에서 최대한 편의를 도모하려고 하였다.

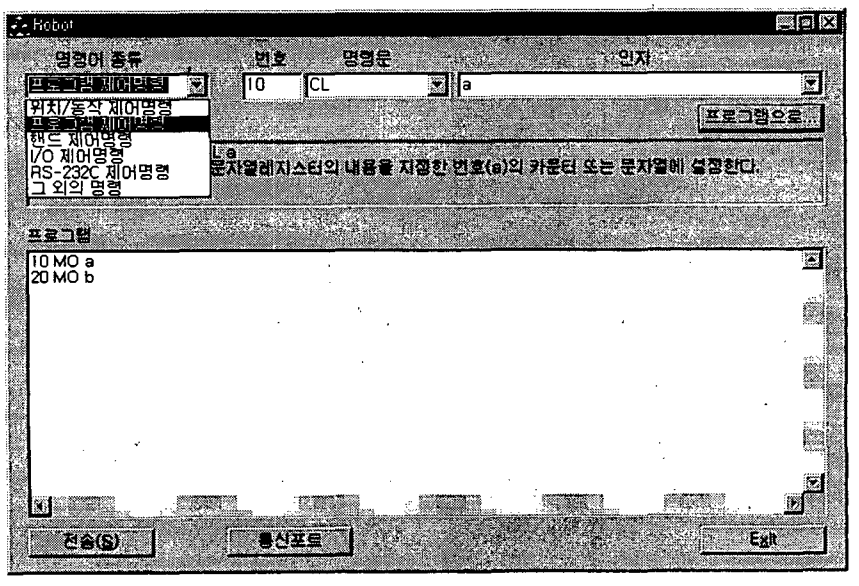


그림 3.18 개발된 Host Program

로봇 Host Program에 Vision System을 통합하였다. 통합하는 이유는 우선, Host Program이 부하가 크지 않아 Vision System과 통합하더라도 Vision처리를 하는데 영향을 받지 않고, Vision System과의 직렬통신에 문제점이 드러났기 때문이다. 분리되었을 때 Robot Host는 직렬포트 2개를 동시에 열어 Vision System과, 그리고 로봇 제어기와 통신을 해야 하는데, 이렇게 포트 2개를 동시에 사용하는데 어려움이 많았고, 통신으로 인한 System Delay가 커져서 전체 성능에 영향을 주었기 때문이다. 따라서 한 시스템에 Vision System과 Robot Host를 통합하였다. Vision System에 메뉴를 하나 추가하여서 이곳에서 Robot Host를 불러내는 형식을 취하였는데, 그림 3.19은 이를 설명한다.

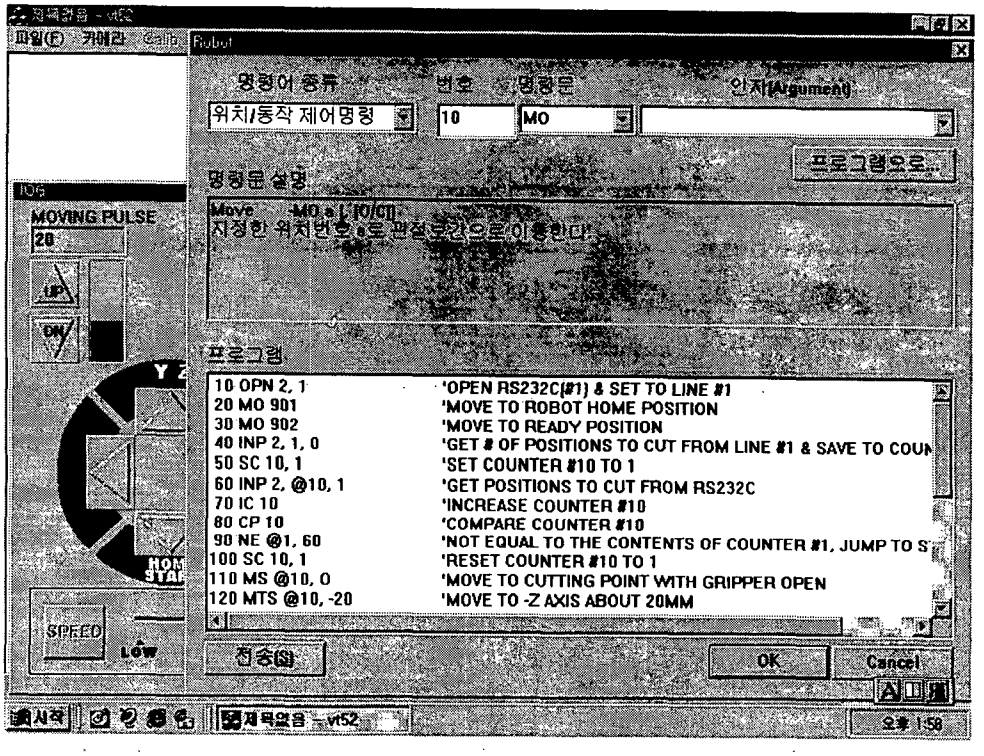


그림 3.19 통합된 Robot Host Program

기본적인 구성은 통신포트 1개를 사용하였으므로, 그리고 사용자는 통신포트에 신경 쓸 일이 없으므로, 화면에서 통신포트 설정에 대한 화면을 없앴다. 사용자는 Host Program에서 필요한 프로그램을 작성하고 이를 로봇 제어기에 다운로드 한 후, 이를 실행한다. 로봇 제어기에서 Vision Data를 받으려면 다음의 문장을 반드시 실행해서 원격으로 Vision Data를 받아야 한다.

```
10 OB +1        'Digital 출력라인 #1을 ASSERT한다.
20 TI 10        '1초동안 Delay
30 OB -1        'Digital 출력라인 #1을 NEGATE한다.
```

그림 3.20 Vision Data Request Program

로봇 제어기에서 제공하는 Parallel I/O부분에서 본 연구에서는 범용출력#1 라인을 사용하여 Host Program과 통신을 한다. 그림 3.21은 로봇제어기의 Parallel I/O를 나타낸다.

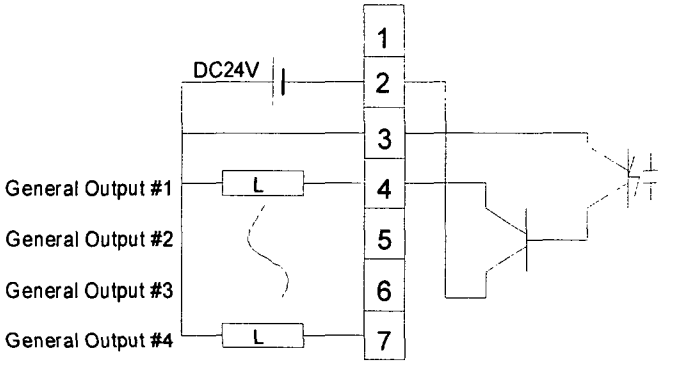


그림 3.21 로봇 제어기의 Parallel I/O

위의 로봇 프로그램은 로봇제어기와 Host Program과의 Digital I/O를 주고받기 위한 문장으로써, 로봇 프로그램에서 필요한 때에 Vision Data를 받기 위해서 Host Program과 위와 같은 방법으로 통신한다. 즉, Digital 출력라인이 ASSERT되면 Host Program에서는 주기적으로 이를 Check하다가 Digital 입력라인이 ASSERT되면 Vision Routine을 실행하여 Vision Data를 구하고, 이를 로봇제어기에 보내주는 역할을 하는 것이다. 다음 그림 3.22는 이를 설명한다.

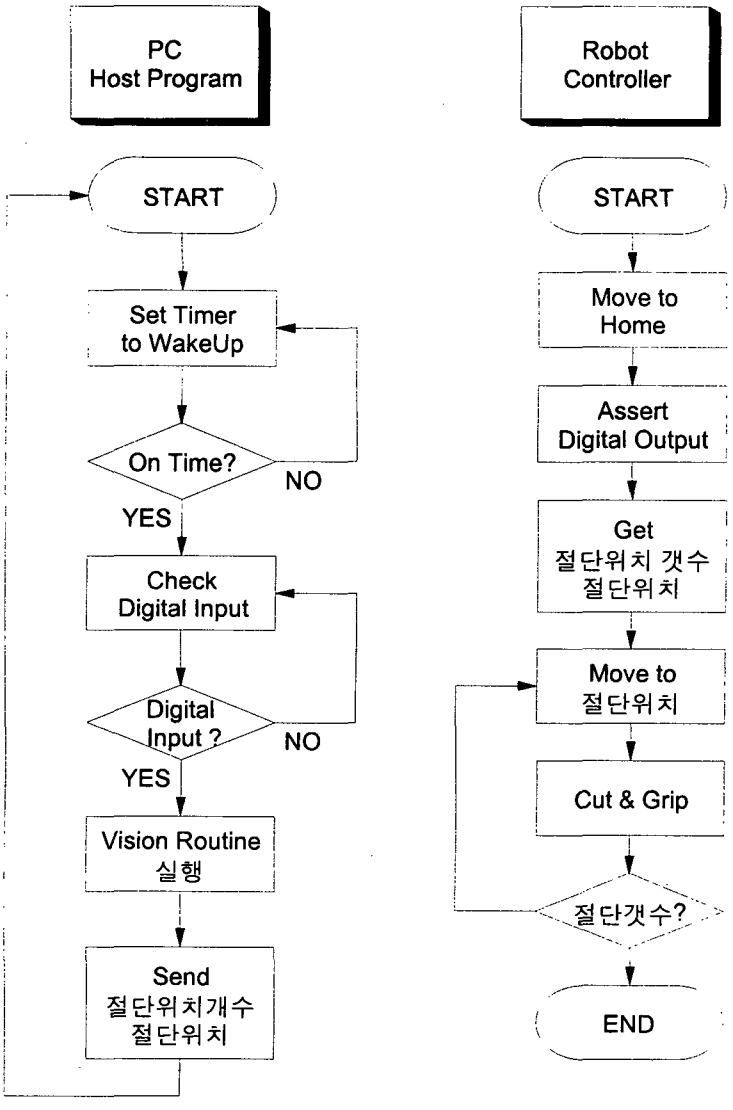


그림 3.22 로봇제어의 흐름도

Vision Data는 Vision Routine에서 Calibration되어 로봇 좌표계에 맞는 데이터를 생성한다. 따라서 이 데이터를 로봇에 전송하면 된다. 다음 Routine은 Host Program과 Robot Program을 나타내는 것인데, 전송과 송신에 대한 핵심 루틴이다.

```
10 SEND("PRN 4");              // Send the number of position data
20 SLEEP(1000);                // Delay about 1 second
30 FOR ( I=1 TO 4 )            // LOOP
40     SEND("PRN x[i],y[i],z[i],a[i],b[i],c[i]");
50 NEXT                        // END of LOOP
```

그림 3.23 Host Program에서의 전송 프로그램

```
10 OPN 2, 1              'Open RS-232C Port #2
20 INP 2, 1, 0           'Get # of position data in counter #1
30 SC 10, 1              'Set Counter #10 to 1
40 INP 2, 20, 1          'Get temp. position data from RS-232C
50 PL @10, 20            'Save to each position number
60 IC 10                 'Increase Counter #10
70 CP 10
80 NE @1,40              'Not EQ. to content of #1, jump to #40
```

그림 3.24 로봇 제어기에서의 수신 프로그램

그림 3.24의 프로그램에서 주의해야 할 점은, PL(Position Load) 명령은 2번째 인자의 위치를 1번째 인자의 위치에 저장하는 문장인데, 이곳에서 간접지정방식(Indirect Method)을 사용할 때, 첫 번째, 두 번째 인자의 위치번호는 미리 교시가 되어야 한다는 점이다. 이는 로봇제어기상의 제한사항이며, 이로 인해 프로그램의 길이가 길어질 수 있다.

그림 3.25는 화면에서 프로그래밍 하는 모습을 보이며, 이를 통해 로봇이 이송을 하는 모습을 보인다. 그림 3.25에서 왼쪽의 화면은 페트리디시를 이동시키는 모터를 구동하는 Jog화면이며, 오른 편에 리얼타임으로 처리하는 Vision System을 보인다. 가운데 주황색으로 굵게 나타나는 곳이 절단해야 할 위치를 표시하고 있으며, 내부적으로 이 좌표를 계산하여 로봇제어기에 전송한다.

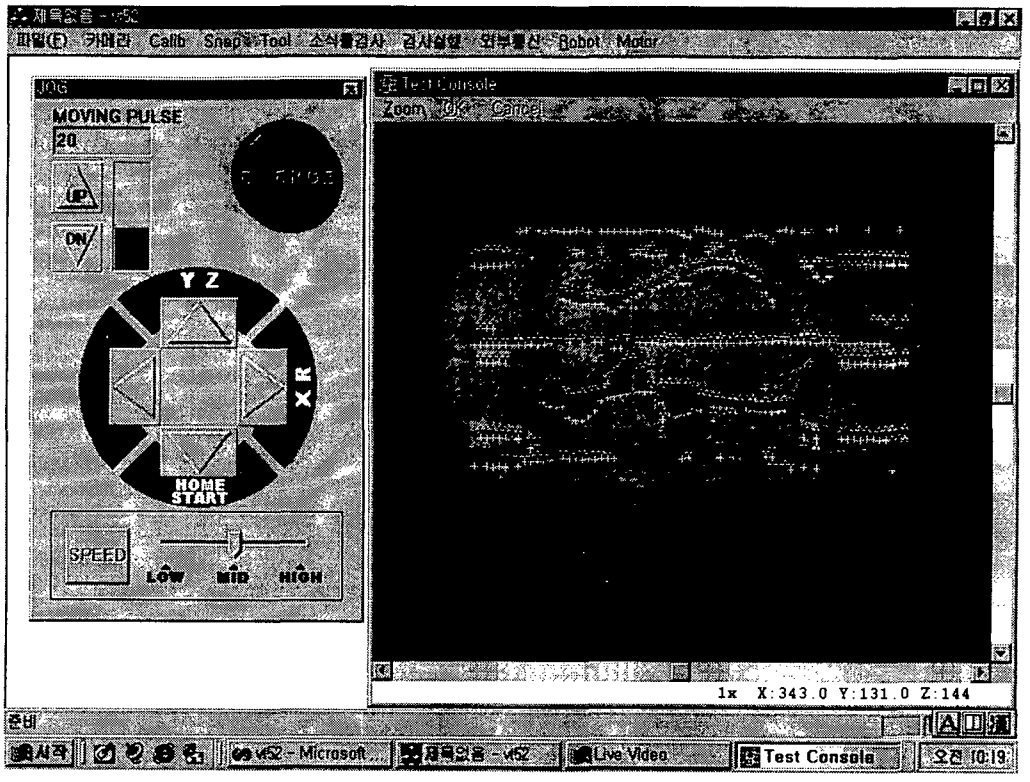


그림 3.25 Vision Data를 처리하는 모습

제 4 절 핸들링 시스템 통합 성능 테스트

1 시스템 구성

성능 테스트는 실제로 사용될 환경과 똑같이 구성을 하여 실제로 사용될 때의 문제점 및 제반사항을 점검하고자 하는 목적으로 이루어 졌다. 실험용으로 제작된 배양접시(Petry Dish)에 생면공학연구소의 도움으로 씨감자를 배양하여 본 실험의 재료.사용하였다. 통합 핸들링 시스템에는 로봇시스템, 그리고 자동으로 절단부위를 추출할 비젼 시스템, 그리고 이 모두를 통합, 제어할 S/W가 있는 PC등이 전체 시스템의 구성이다.

1) 페트리 디시 핸들링 시스템

두 개의 Step Motor로 X/Z축을 제어하는 XZ 로봇 시스템과 Gripper, 그리고 페트리 디시를 이송하여 Loading/Unloading 하는 공압 Stage로 이루어져 있다. 성능 테스트에서는 시스템 안전성을 위해 각 스테이지의 동작마다 약간의 시간적인 지연과 동작 속도를 낮추어서 실행하였다. 따라서 성능 테스트에서의 결과로 나온 시간은 최적 운영 상태의 동작시간이 아닌 것임을 밝힌다.

2) 로봇 시스템

Vision System과의 원활한 통신을 위하여 약간의 시간 지연이 있다. 로봇은 Vision System에서 받은 위치데이터를 기초로 하여 소식물체를 절단하여 배출 Tray에 절단된 식물체를 옮겨 놓는 역할을 한다.

3) Vision System

페트리 디시 핸들링 시스템이 Vision 처리를 할 수 있도록 소식물체가 담긴 페트리 디시의 뚜껑을 개방하고 Back Light Stage에 페트리 디시를 가져다 놓으면, Vision System이 실행된다. 처리결과는 절단할 위치갯수, 그리고 절단위치이며, 이 정보는 RS-232C를 통해 로봇제어기로 넘겨진다. Vision System에서 처리된 위치정보는 실제 Back Light Stage의 중심을 기준으로 Calibration 된 정보이며, 로봇제어기에 넘겨지기 이전에 로봇 좌표계로 변환되어 넘겨진다.

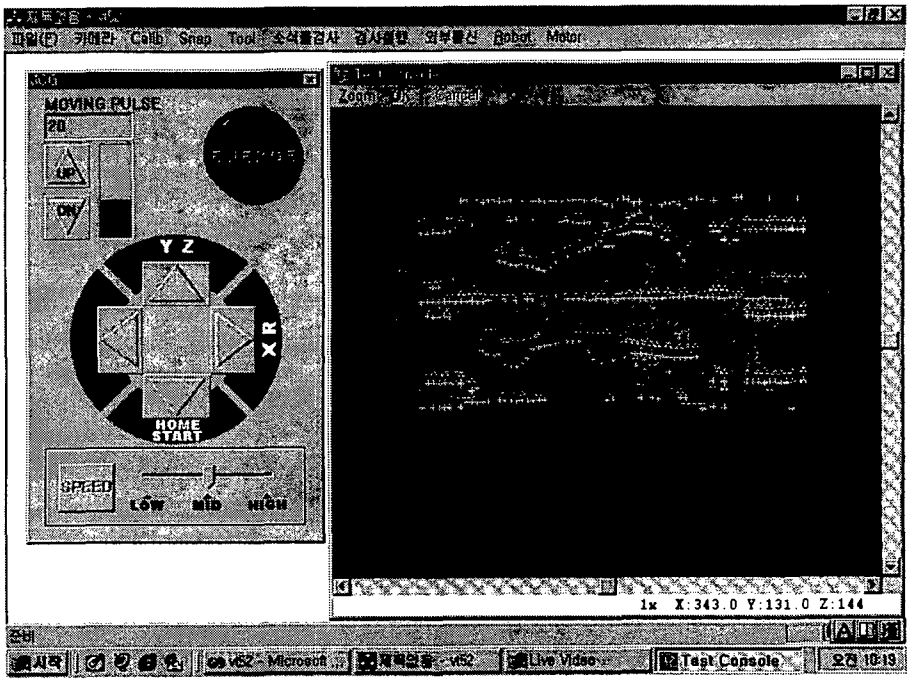


그림 3.26 조그 화면

4) 통합제어 시스템

PC에서 구현된 통합제어 시스템은 주변기기, 즉 핸들링 시스템, 로

봇 시스템, Vision System을 통합, 제어한다. 핸들링 시스템은 로봇과 마찬가지로 교시를 통해서 각 위치를 결정하며, 이를 위해 Jog화면을 사용한다. 그림 3.26은 Jog화면을 나타낸다. Jog화면을 통해 필요한 위치를 교시하며, 이를 등록하는 화면이 그림 3.27에 나타나 있다.

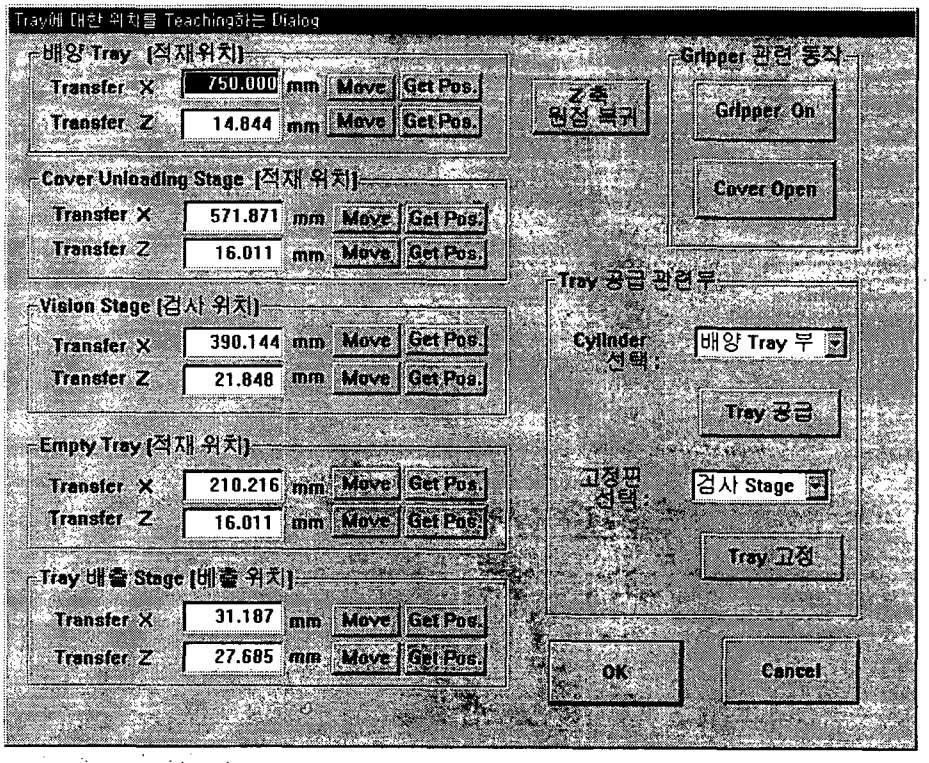


그림 3.27 교시한 위치데이터를 등록하는 화면

이처럼 교시된 위치데이터를 기준으로 각 스테이지는 움직이며, 각 동작은 적절한 함수로 표현되어 있다. 이러한 함수들은 메인 메뉴에서 시험할 수 있다.

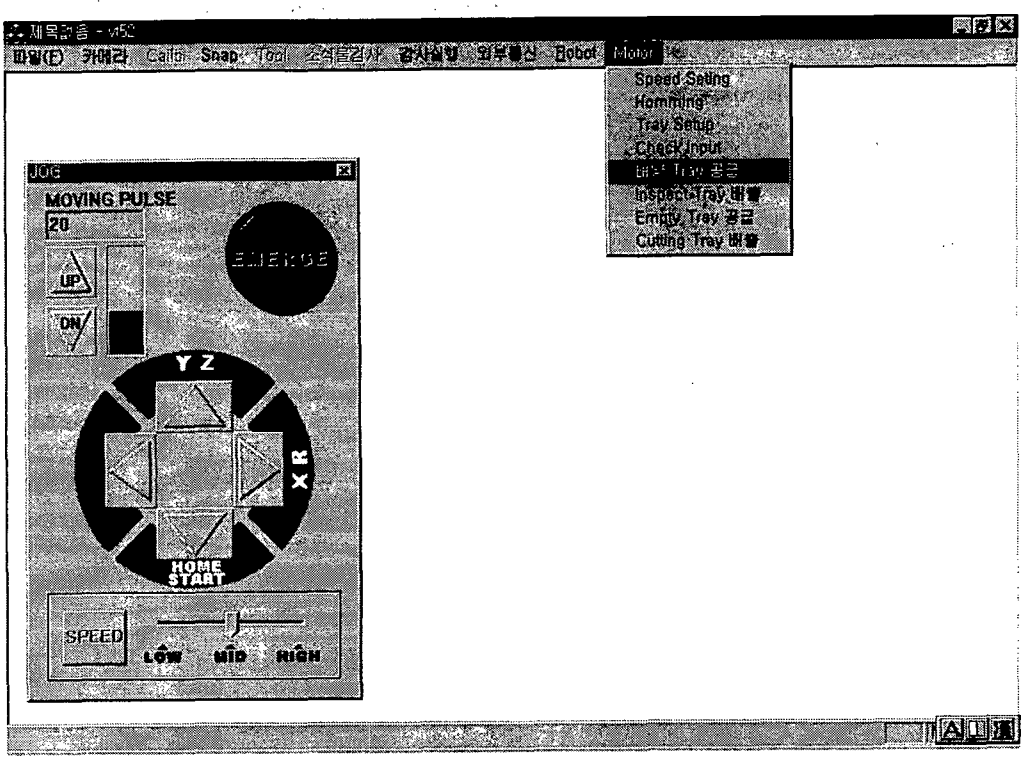


그림 3.28 동작함수 메뉴

Robot System과는 Digital I/O로 통신하며 필요한 위치데이터는 RS-232C를 통하여 주고 받는다. 또한 프로그램을 직접 작성하여 Download할 수 도 있다. Movemaster Programming을 Teach Pendant에서 하기에는 너무 힘들므로, 그림 3.29와 같이 PC를 이용하면 매우 편리하다.

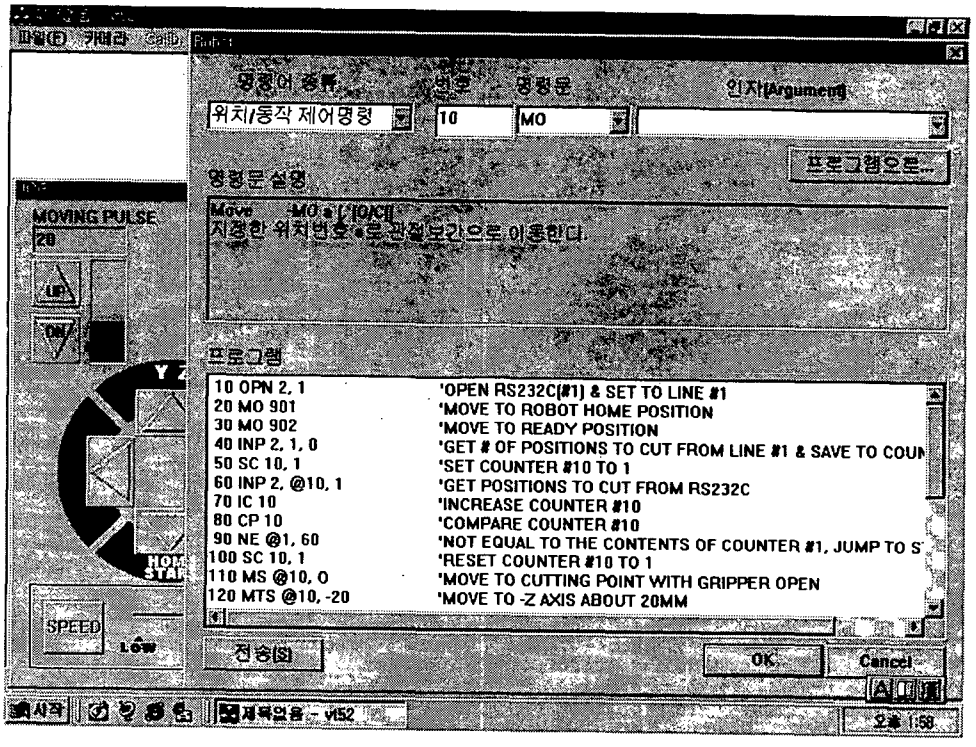


그림 3.29 Robot Programming을 하는 모습

Vision System은 사용자가 테스트를 위해 Manual로 검사할 수 있고, 프로그램상에서 자동적으로 함수를 Call하여 사용할 수도 있다. 그림 3.30은 자동으로 테스트하는 그림이다.

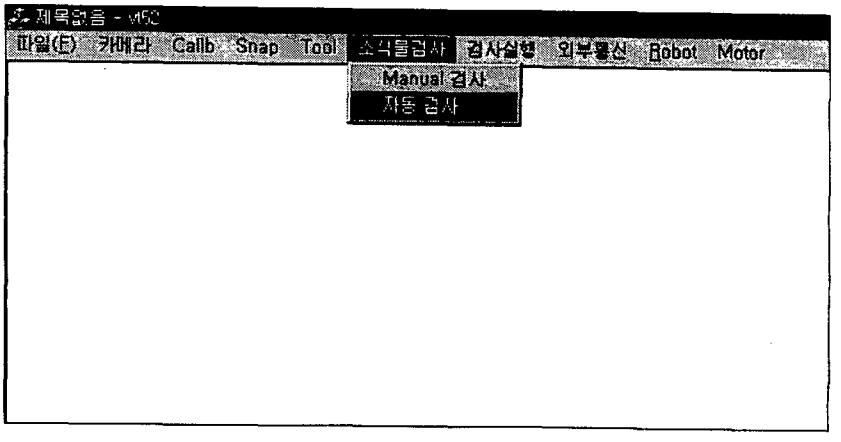


그림 3.30 Vision System을 자동으로 검사

2. 주요 단위시스템 성능실험

핸들링 시스템 전체를 통합을 한 후 주요 단위시스템의 기능에 대한 성능을 실험하였다.

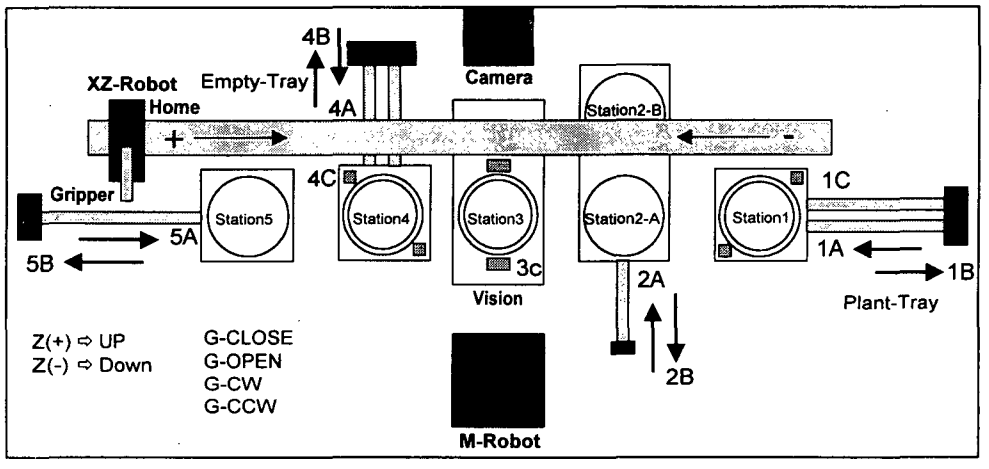


그림 3.31 로봇 핸들링 시스템 구성

1) XZ Robot, M-Robot, Gripper Home Position

그림 3.31의 XZ Robot을 Home 위치로 이동시키고, Gripper를 Close한다. M-Robot은 RV-E2 로봇을 지칭하며, Vision System의 Camera를 방해하지 않는 사용자 정의의 Home Position으로 이동시킨다.

2) 배양 Tray 공급

배양된 소식물체는 페트리 디시에 담겨져 있으며 페트리 디시는 뚜껑이 닫혀있다. Station 1에서 공급된 페트리 디시는 Station 3로 옮겨진 후 뚜껑은 Station 2A에 놓인다. 뚜껑은 Station 2A에서 Station 2B로 옮겨진 후 배출된다.

3) Empty Tray 공급

Station 5에 있는 빈 페트리 디시가 Station 4에서 뚜껑이 열리고 뚜껑은 Station 5에, 빈 페트리 디시는 Station 4에 위치하게 된다. 절단된 소식물체는 Station 4에 있는 빈 페트리 디시에 옮겨지게 된다.

4) Vision Processing/Robot Cut-Pick & Place작업

2), 3)의 페트리 디시가 모두 지정된 위치에 도달하면 통합제어 시스템은 Vision Processing을 시작한다. Vision Processing을 하기 전에 반드시 카메라의 모드가 선택되어 있어야 하며, Processing이 끝나면 절단해야 할 위치의 개수와 위치 데이터가 로봇 제어기에 전송된다. 로봇은 위치 데이터가 전송되길 대기하고 있다가 데이터가 전송이 되면 곧바로 작업을 시작한다. 즉, 절단한 위치로 이송을 한 후, -Z축 방향으로 절단할 위치까지 하강 후, 절단 및 Grip을 행한 후, 절단된 소식물체를 Stage 4에 위치한 페트리 디시에 옮겨 담는 작업을 시작하는 것이다. 이 작업은 모든 절단해야 할 위치 개수만큼 반복되며, 작업이 끝났을 때에는 통합시스템에 작업이 끝났음을 알린다.

5) Inspect Tray 배출

Vision Processing 및 로봇의 작업이 끝나면 Station 3에 있는 페트리 디시는 Station 2로 옮겨져 배출된다. 뚜껑은 미리 배출되었으므로, 바로 배출된다.

6) Empty Tray 배출

절단된 소식물체가 들어있는 Station 4의 페트리 디시는 Station 5에 있는 뚜껑을 가져와 Station4의 페트리 디시와 결합한 후, Station 5에 옮겨져 배출된다.

3. 성능 테스트 결과

본 실험은 각 단위 시스템의 동작 기능의 적합성을 알아보는데 중점을 두었고, 시스템의 안전성을 위해 속도를 낮추어 한 사이클 작업시간을 1분 정도에 맞추었으나, 시스템의 안정성이 확보되는 한계 내에서 최대한으로 속도를 높일 수 있다. 종합적으로 전체 시스템의 목표 기능 달성에 큰 문제점이 없는 것으로 나타났다.

1) 배양 Petri Dish 공급

배양액에 담겨져 있는 소식물체가 페트리 디시에 담겨져 공급된다. 페트리 디시 핸들링 그리퍼와 XZ 로봇 시스템에 의해 정확한 공급과 Loading/Unloading 동작이 잘 이루어지고 있다. 특히 디시 뚜껑의 개폐 동작이 잘 이루어지고 있다.

2) Empty Petri Dish 공급

절단된 소식물체가 담겨질 빈 페트리 디시를 공급하는 과정이다. 역시 정확하게 교시된 위치로 페트리 디시의 뚜껑 개폐 동작이 잘 이루어지고 있다.

3) Vision Processing에 의한 Cutting-Gripping & Place 작업

Vision System에 의한 처리시간은 거의 실시간으로 처리가 이루어지고 있으며, 그리핑 및 절단할 위치 데이터가 로봇제어기에 전송이 잘 이루어지고 있다. 전송되는 즉시 로봇은 절단할 위치로 이동하여 절단작업을 수행하게 된다. Gripping-Cutting 동작의 구동원을 사용하기 때문에 약간의 시간을 잡아먹고 있지만 실시간 처리에는 문제가 없다. Place(이식) 동작의 시간은 로봇의 최대 속도를 이용하면 단축이 가능하다. Gripper &

Cutter 메카니즘 제작과정에서 Cutter의 고정하는 부분이 약간 헐겁게 고정되어 카터의 위치가 동작시 약간의 위치 오차가 생기나 로봇 동작의 Offset 개념으로 처리가 가능하다.

4) Inspect Petri Dish 배출

작업이 끝난 페트리 디시는 뚜껑이 열린 채로 배출된다. 실험 결과 성공적으로 실행되었다.

5) Empty Petri Dish 배출

Vision 및 Robot 작업이 끝나면 즉시 절단된 소식물체가 담긴 Empty Tray가 배출된다. Inspect Tray 배출작업과 거의 동시에 이루어진다.

이상과 같이 본 개발시스템에 대한 실제 작업 적용 실험을 통해, 본 과제의 목표하고 실제 현장에 적용할 수 있는 소식물체 인식도, 파지, 절단 위치 정밀도, 위치제어 속도, 페트리 디시의 Loading/Unloading 동작, 페트리 디시 뚜껑의 개폐동작 등에 대한 시스템 기능 및 성능을 충분히 구현할 수 있었다. 본 개발 시스템은 Prototype 시스템이므로 실제 현장에 그대로 투입하기에는 부족한 점이 많다. 페트리 디시의 투입 및 배출 위치, 스테이션 위치, 관련 주변 시스템과 연관성, 조직배양실 운영 등을 고려하여 충분한 기능 및 성능을 발휘할 수 있도록 시스템 구성을 최적화 하여야 한다. 특히, 조직배양실에서 여러 가지 오염원을 배제할 수 있는 시스템, 즉 청정 작업을 위한 시스템 구성 및 기구부의 형태에 대해 추가로 연구되어야 한다. 생산성 향상을 위한 구동부의 속도 증가는 시스템의 비용과 상관 관계가 있으므로 적절한 성능을 갖는 시스템을 선택하여야 한다.

제 4 장 개발 시스템에 대한 경제성 평가

제 1 절 경제성 평가에 있어서 원가계산

1. 원가의 구성

여기서 취급하는 원가계산은 프로젝트의 경제성 평가에 필요한 제품의 예상원가의 계산 방법이지, 회계학상의 엄밀한 의미에서의 원가계산은 아니다. 프로젝트의 경제성평가는 한마디로 제품의 판매에 의한 수익성을 여러 가지 관점에서 평가하는 것이다. 따라서 경제성 평가에 있어서 가장 중요하고 기본적인 요인은 제품의 원가와 판매가격이다. 또한 이 원가와 판매가격을 전제조건이 미확정된 시점에서 예상해야 하므로 경제성 평가에 있어서는 이들 요인의 정도(精度), 다시 말해 경제성 평가의 정도에 대한 한계를 알아두는 것이 절대로 필요하다. 그러기 위해서는 우선 원가의 계산 방법 뿐만 아니라 원가를 구성하는 요소가 변동되면 원가에 어떤 영향을 주는가 하는 것도 관심을 가져야 할 것이다.

우선 원가를 구성하는 요소를 표 4.1에 나타내고 다음에 각 요소에 대해서 알아본다.

표 4.1 원가의 구성

총원가	제조원가	변동비	원료비
		고정비	노무비 경 비 부동산 임대료 감가상각비 보험료 특허권 사용료 수선비 고정 자산세 공장 관리비
	판매관리비 판매비 일반 관리비		
	이 자 건설자금 운전자금		

2. 원가 구성 요소에 대한 설명

(1) 원료비(재료비)

여기서 말하는 원료 중에는 주원료 외에 부원료, Utility(용수, 전기, Steam)도 포함한다. 이들의 필요량을 통상 Process에 의해서 기술적으로 결정되어 제품의 단위량(통상은 제품1톤)당 필요량을 원단위 (原單位, Consumption Figure)라고 칭한다. 따라서 원료비는 식(4.1)에 의해 구해진다.

$$\text{원료비} = \text{원단위} \times \text{원료단가} \times \text{생산량} \qquad (4.1)$$

식(4.1)에서 생산량은 '설비용량×가동률' 로 표시되므로 다음과 같이 계산할 수도 있다.

$$\text{원료비} = \text{원단위} \times \text{원료단가} \times \text{설비용량} \times \text{가동률} \qquad (4.2)$$

이상과 같이 원료비는 Plant의 가동률에 의해 변동되므로 변동비(Variable Cost)라고 불리어진다. 이에 반해 노무비, 경비등은 Plant의 가동률과 관계없이 비용은 고정되어 있으므로 고정비(Fixed Cost)라고 불리어진다.

기업의 해외진출의 주된 동기로서 원료의 확보가 거론되는데, 이것은 원료비가 Project의 경제성을 좌우하는 중요한 인자임을 의미한다. 특히 화학공업에 있어서는 거의 원료비가 경제성을 결정한다해도 과언이 아니다.

따라서 Project 기획에서는 원료비의 산정은 신중히 행해져야 한다. 또 원료비가 변동하기 쉽다는 것, 그 변동이 경제성에 크게 영향을 미친다는 것을 고려한다면, 원료비가 변동할 때의 경제성에 주는 영향도 정량적으로 파악해두는 것이 필요하다. 이와 같이 경제성을 좌우하는 요소가 변동할 때 경제성에 주는 영향을 검토하는 것을 감도분석(感度分析, Sensitivity Analysis)이라고 불리어 지는데, 원료비에 대한 감도분석은 매우 중요하다.

원료를 그대로 설비에 투입하지 않고, 한번 외부에 발주하여 가공시킨 후에 사용하는 경우가 있는데, 이 외주가공비도 원료비중에 포함시키는 것이 일반적이지만, 이 비용을 경비로서 처리하는 수도 있다.

(2) 노무비

노무비에는 공장에서 생산에 직접 종사하는 노무자의 비용이 포함된다. 노무자에게 지급하는 임금 제수당 외에 산업재해 보험료, 복리 후생비 등도 노무비에 포함된다. 노무자의 직능별 인원수는 인원수를 기점으로 해서 현지 노무자의 작업효율에 의해 조정함으로써 구해진다. 이렇게 구해진 인원수에 현지인의 노무비 단가를 곱하면 된다.

위에서 말한 바와 같이 이 노무비 단가에는 노무자에 주는 지급 임금 총액 외에 회사가 부담하는 회사보험료, 복리후생비 등의 비용(Fringe Benefit; 附加給付)도 포함되는데 이 Fringe Benefit을 노무비에 포함시키지 않고 경비로서 처리하는 경우도 있다. 어떠한 경우에도 원가로서는 같다. 지급 임금 총액에 대한 Fringe Benefit의 비율은 대개 10~20 %이다.

화학공업에서는 Operator의 수가 매우 적어서 노무비의 제조원가에서 차지하는 비율은 적다. 부언한다면 화학공업의 해외입지 동기는 노동력 확보는 아니다. 한편 선진기업의 진출을 받아들이는 개발도상국 측에서는 고용기회의 확대만을 노릴 수 있는 노동 집약적 산업의 수입을 희망하는 곳이 많다. 예를 들면, 70년대 우리 정부는 마산 수출자유지역에의 외자도입 때에 100평당 20인의 고용을 기준으로 하고 있다. 이런 경우에는 진출업종이 필연적으로 섬유산업, 전기산업 등 노동 집약산업에 집약된다. 이와 같은 현지 측의 사정도 고려해서 노동 절약적인 화학공업 등에서도 가능한 한 고용을 증대시키려는 노력을 해야 할 것이다. 원래 노무비가 차지하는 비율이 적으므로 저렴한 현지인 노동력을 다소 증대시켜도 제조원가에 주는 영향은 무시될 수 있다.

(3) 경비

원료비, 노무비 외에 제조에 필요한 비용을 일괄해서 경비라 부르

는데 다음과 같은 비용이 포함된다.

① 부동산임대료

토지, 건물 등의 부동산을 구입하는 경우는 자산이 증가하는 것이므로 이 비용은 자본적 지출로서 처리된다. 이 자산 중 상각자산(償却資産)에 대해서는 원가상각비의 형태로 원가를 구성한다. 이에 대해서 이들 부동산을 임대하는 경우는 자산이 되지 않으므로 경비로 처리해야 한다.

② 감가상각비

토지, 건물, 설비등과 같이 반복해서 사용할 수 있는 자산을 고정자산(Fixed Assets)라 부르고, 이 고정자산중 사용에 따라 그 기능 및 본체가 소모하여 가치가 감소해 가는것에 대해서는 그 가치의 감가액을 추정하여 계산해 경비로 처리해야 한다. 이 비용을 감가상각비라 한다. 감가상각비에 대해서는 다음에 상세하게 서술하므로 참조하기 바란다.

③ 보험료

Plant에 대해 부보(付保)하는 상해보험과 배상 책임보험의 Cost이고 총건설비의 0.5~1 % 정도이다.

④ 특허권 사용료

생산실적에 관계없이 일정금액을 지불하는 Paid Up의 경우, 생산량에 따라서 지불하는 Running의 경우, 또는 양쪽을 조합시키는 경우의 3가지 지불방법이 있다. 화학 Process의 경우 Running Base로 제조원가의 2~6 % 정도이다.

⑤ 수선비

연간 수선비는 대개 총 건설비의 3 % 정도인데 Plant의 연수가 증가함에 따라 수선비가 많이 든다. 따라서 수선비는 매년 체증하여 계산하는 것이 보통이다. 그러나 Plant의 사용연수를 연장시키려고 수선을 할 경우 이 비용을 비용지출이라고 인정되지 않고 고정자산의 증가로 되는 때문에 자본적 지출로 간주한다. 고정자산의 증액분은 감가상각비의 형태에서만 비용으로 산입된다.

⑥ 고정자산세

세율은 공장을 건설하는 나라 또는 지방에 따라서 다르므로 미리 세율을 조사해야만 하지만, 대개 총 건설비의 1~2 % 정도라고 보는 것이 좋다.

⑦ 공장관리비

공장관리비에는 공장에서 직접 제조에 관계하지 않는 노동을 하는 자의 임금, 직장/기술원/사무직원 등의 급료, 공장 소모품비, 공구/기구품비, 그 외 간접적인 경비가 포함된다. 이들 비용은 직접 노무비의 100~200 % 정도이다.

(4) 판매비, 일반 관리비

판매비, 일반 관리비는 직접 제조에는 관계없는 비용이므로 제조원가에는 포함되지 않는다. 즉 엄밀한 의미에서의 원가계산은 판매비, 일반 관리비에 대해서는 행해지지 않는다. 실제로 이들 비용을 제품의 가격에 분산시켜 산정키 위해서는 제조원가, 또는 판매가격을 기초로 한 가격비율로 나누어 부과하는 방법 외에는 없기 때문에 원가로서의 정확성은 갖추지 못

하게 된다. 다만 모든 비용이 다 원가로서 인정되지 않는 것은 아니고 예를 들면 포장, 발송비 등은 제품에 직접 관련되므로 제조원가로서 처리 될 수도 있다.

판매비, 일반 관리비에 포함되는 것은 공장 외, 즉, 본사부문의 경비(영업비, 개발연구비 포함) 및 인건비이다. 위에서 말한 대로 이들 비용은 제조원가 또는 판매가격의 Percentage로 구하는 것이므로 이 견적은 자사의 본국에서의 Percentage에 진출국에서의 경향을 가미하여 적당히 판단해야 한다. 대체적인 안목으로서는 판매원가의 10 % 정도이다.

(5) 이자

이자 등 금전상의 비용은 재무계산에서는 영업외 비용이 되고, 또 원가계산상에서도 원가로 간주되지 않는다. 그러나 세금이 공제된다는 의미 때문에 비용이라는 것은 변함이 없어 이자를 총원가 중에 포함시키고 있다. 이자는 건설자금용의 장기 차입금과 운전자금용의 단기 차입금으로 대별된다.

차입금액을 결정하는 것은 총 투자액의 자금조달원천을 결정하는 것이다. 일반적으로 자금의 조달원천은 다음과 같이 분류된다.

① 내부원천

(i) 사내유보(社內留保)
(ii) 감가상각

② 외부원천

(i) 자기자본 : 주식발행(자본금)
(ii) 타인자본 : 차입금, 회사채

원가계산에서는 타인자본의 Cost를 비용으로 취급하고 자기자본에서는 Cost, 즉 이자를 지불하지 않는 이유 때문에 비용으로 일체 인정되지 않는다. 자기자본을 타인에게 대부한다면, 당연히 이자를 받으므로 신규 Project의 기획에서는 자기자금에도 이자를 수취하는 기회를 잃고 있다는 의미에서 Cost가 관계되어 있다는 것을 염두에 두어 경제성 평가를 행해야 한다. 이점에 대해서는 다음에 상세하게 설명되어 있다.

3. 감가상각비

(1) 개 요

Plant 건조물 등의 고정자산은 사용연수가 있어 연수가 지나면 사용할 수 없게 된다. 이들 설비는 매년 사용함에 따라 그 기능 및 본체를 소모시켜 그만큼 설비가액을 상실한다. 따라서 매년도의 고정 자산평가에서는 그 연도의 감가액을 자산의 장부가액으로부터 공제해야 한다. 이 감가액을 감가상각비(Depreciation Cost)라 한다. 감가상각비는 단순히 고정자산의 감가액일뿐 아니라 원료비, 노무비 등과 같이 제조원가로서 취급된다.

설비의 기능 및 본체의 소모는 눈으로 보이지 않기 때문에 실지검사(實地檢査)로는 측정할 수 없다. 따라서 감가상각에는 장부상의 추정계산에 의해 구해지지 않을 수 없다. 이 추정계산의 결과가 바른가 어떤가에 대해서는 실제로 증명할 수 없다. 감가상각비의 계산은 기업의 의사에 따라 좌우된다. 감가상각비는 세금의 공제액으로서의 의미를 지니지만 이 계산이 기업의 의지에 따라 좌우된다면 세금의 공평부담 원칙으로 보아 좋지 못하므로, 다음에 설명하는 바와 같이 감가상각비 계산의 기본적인 Rule이 결정되어 있다.

(2) 사용연수

우선 실비의 내용연수에 대해서는 한국에서는 법인세법에 결정되어 있는 법정이용 연수표에 따라야 한다. 미국에서는 표준내용 연수표에 지정되어 있다. 이와 같이 이용 연수는 각국의 법령으로 결정되어 있으므로 우선 그것을 조사해야 한다.

(3) 감가상각방법

감가상각 방법은 정액법, 정율법, 연수합계법, 감채기금법, 생산량비례법 등이 있지만, 그 중에서 모든 유형, 무형 고정자산과 이연자산(광업권 포함)에 적용되며 계산이 간략한 정액법에 대해 알아본다.

정액법(Straight Line Depreciation)은 매년도에 정액의 상각비를 계상하는 방법이다. 상각액은 다음과 같이 구한다.

$$\text{상각액} = \frac{\text{초기투자비} - \text{잔존가액}}{\text{이용년수}} \qquad (4.3)$$

즉, 상각자산의 원가를 1000, 잔존가액을 100, 이용연수를 10년이라 하면,

상각비 = (1000 - 100) / 10 = 90
상각율 = 상각액/초기투자비 = 90/1000 = 9 % 이다.

상기의 예에서 감가상각액과 미상각잔고를 그래프에 나타내면 그림 4.1과 같이 되는데 이들이 직선으로 나타나므로 미국에서는 정액법을 직선법(Straight Line Depreciation)이라고 한다.

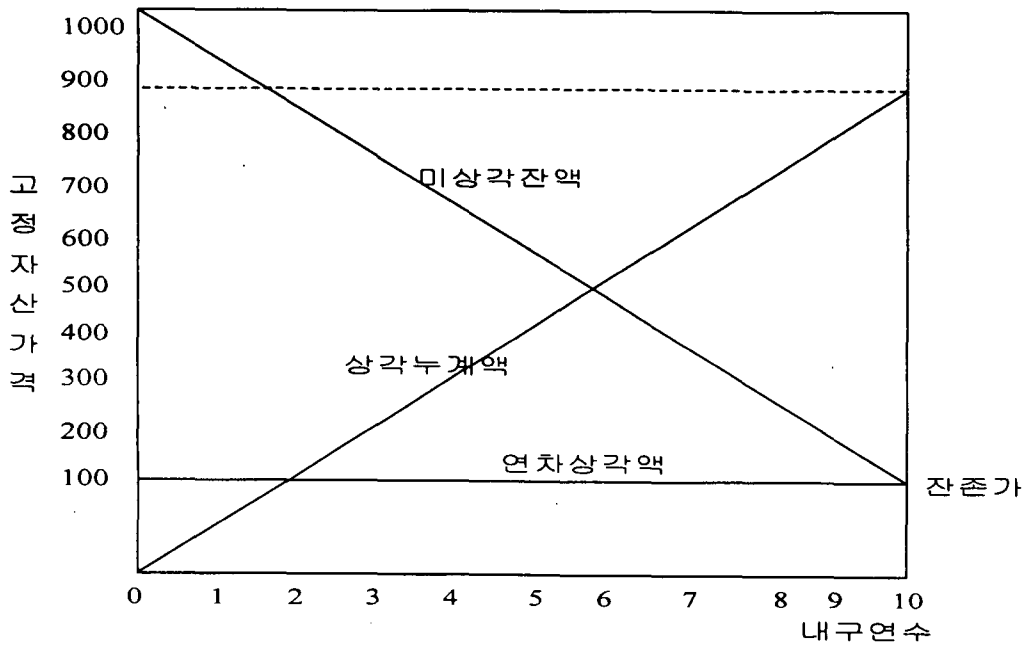


그림 4.1 정액 감가상각법의 예

제 2 절 DCF에 의한 Project의 경제성 평가

1. 자금의 시간적 가치

Plant 건설 Project는 착공으로부터 완공까지 통상 2~4 년이 걸리고, 기획단계를 포함하면 짧은 경우에도 5~10 년 정도의 장기간이 걸리는 경우도 허다하다. 이 장기간에 대량의 투자가 행해지고 그로부터 다시 장기간 공장을 운전하여 제품을 판매하고 투자한 자금을 회수하면서 이익을 얻는 것이다. 이렇게 장기간에 걸친 대량의 자금유입을 수반하는 프로젝트의 경제성을 평가하는데는 외견상의 자금량만을 단순히 비교해서는 안되며, 그 시간적 차이도 고려하여 평가할 필요성이 있다.

예를 들어 현재 수중에 있는 100만원과 2년 후에 입금이 예정되어 있는 120만원 중 어느 것이 가치가 있는가를 판정해 보면, 외견상의 금액으로는 확실히 2년 후의 120만원이 크지만 시점이 다르므로 이대로 비교할 수 없다. 비교하기 위해서는 어떠한 시점으로 같게 만들어야 한다. 현재의 100만원은 은행에 예금하면 이자가 생기고 투자를 하면 이익을 얻을 수 있다. 따라서 현재의 100만원은 2년 후에는 $(100+\alpha)$만원의 가치를 지니게 되며, 동일한 방법으로 2년 후에 입금이 예정되어 있는 120만원을 미리 빌리려고 하여 이자를 공제하면 $(120-\beta)$만원밖에 손에 넣을 수 없다. 따라서 위의 두 가지 금액의 가치를 비교하는 경우에는 현재의 가치로서 100만원과 $(120-\beta)$만원을 비교하던가 또는 2년 후의 가치로서 $(100+\alpha)$만원과 120만원을 비교해야 한다. 이에 현재의 가치와 장래의 가치간의 일반적인 관계식을 복리공식의 사용으로 구해본다.

현재의 가치를 현가(Present Worth/Present Value)라 하여 P라 하고, 장래의 가치를 미래(Future Value)라고 하여 F로 하자. P와 F의 관계는 [초년도 초에 P를 예금할 때 년이율 i로서 n년말의 원리합계가 F이다]

라고 나타내어진다. 이를 식으로 나타내면,

$$F = P(1+i)^n \tag{4.4}$$

또는 식 (4.4)를 변경하여

$$P = F \times \frac{1}{(1+i)^n} \tag{4.5}$$

여기서 식 (4.4), (4.5) 중

$f_i = (1+i)^n$: 복리계수(Compound Interest Factor)

$f_a = 1/(1+i)^n$: 할인계수

(Discount Factor)/현가계수(Present Worth Factor)

라고 정의한다.

위의 공식에 의해 어떤 시점에서의 일괄 금액은 타시점에서의 일괄 금액표시의 가치로 전환된다.

다음에 매년도 등액의 지출이 있을 때 이들의 장래에 있어서의 일괄금액표시의 가치가 어떻게 되는가를 조사해 본다. 매년도 등액의 지출액을 연불액(Equal Annual Payment)라 하고 이것을 A라 하면, 상기의 관계는 다음과 같이 바꾸어 말할 수 있다. 즉, "매년 말에 A를 예금한 경우 년이율 i로써 n년 말의 원리합계가 F로 된다." 이것을 식으로 표시하면

초년도말의 예금액 : A

2년말의 원리합계 : $A(1+i)+A$

3년말의 원리합계 : $A(1+i)^2+A(1+i)+A$

n년말의 원리합계 :

$$F=A(1+i)^{n-1}+A(1+i)^{n-2}+\ldots+A \tag{4.6}$$

식(4.6)의 양변에 (1+i)를 곱하면

$$F(1+i)=A(1+i)^n+A(1+i)^{n-1}+\ldots+A(1+i) \tag{4.7}$$

식(4.7)에서 (4.6)을 빼면,

$$Fi=A(1+i)^n-A$$

위 식을 참조해서,

$$F=A\left[\frac{(1+i)^n-1}{i}\right] \tag{4.8}$$

$$A=F\left[\frac{i}{(1+i)^n-1}\right] \tag{4.9}$$

식 (4.4), (4.5), (4.8), (4.9)로부터 다시 P와 A간의 다음 관계식을 유도할 수 있다.

$$P=A\left[\frac{(1+i)^n-1}{i(1+i)^n}\right] \tag{4.10}$$

$$A=P\left[\frac{i(1+i)^n}{(1+i)^n-1}\right] \tag{4.11}$$

상기 식에서

$$(F/A, i, n) = \frac{i}{(1+i)^n - 1}$$

: 연불액미래가계수(Annuity Future Worth Factor)

$$(P/A, i, n) = \frac{i(1+i)^n}{(1+i)^n - 1}$$

: 연불액현가계수(Annuity Present Worth Factor)

로 정의된다.

이에 의해 P, F, A 상호간의 관계식이 구해지므로 이것을 표 4.2에 정리하였다. 각각의 계수치는 복리공식 계수표를 참조해야 한다.

표 4.2 복리공식 계수표

식(4.4)	P→F	$F = P(1+i)^n$ $= P \cdot (F/P, i, n)$ $(F/P, i, n)$: 복리계수
식(4.5)	F→P	$P = F\frac{1}{(1+i)^n}$ $= F \cdot (P/F, i, n)$ $(P/F, i, n)$: 할인계수
식(4.8)	A→F	$F = A\left[\frac{(1+i)^n - 1}{i}\right]$ $= A(F/A, i, n)$ $(F/A, i, n)$: 연불액미래가계수
식(4.9)	F→A	$A = F\left[\frac{i}{(1+i)^n - 1}\right]$ $= F(A/F, i, n)$ $(A/F, i, n)$: 감채기금계수
식(4.10)	A→P	$P = A\left[\frac{(1+i)^n - 1}{i(1+i)^n}\right]$ $= A(P/A, i, n)$ $(P/A, i, n)$: 연불액현가계수
식(4.11)	P→A	$A = P\left[\frac{i(1+i)^n}{(1+i)^n - 1}\right]$ $= P \cdot (A/P, i, n)$ $(A/P, i, n)$: 자본회수계수

표 4.2의 공식을 사용하여 투자 프로젝트의 장기간에 걸친 자금의 출입이 단순한 1개의 가치로 전환된다. 현가(P), 종가(F), 연차등가(A)중에서 어떤 것으로 전환해도 좋지만, 2개의 투자 프로젝트의 경제성을 비교할 때는 양 프로젝트에 관해서 동일한 기준의 가치, 예를 들면 현가와 현가를 비교해야만 한다. 프로젝트의 경제성 평가에서는 대개의 경우 현가를 사용한다. 이것은 계산이 간편한 외에도 경제성 평가는 현시점에서의 평가이므로 현가법이 직관적으로 이해하기 쉬운 사정 때문이기도 하다. 앞으로의 경제성 평가에 대한 구체적인 수법도 모두 현가법에 근거하고 있다.

2. Cash Flow

Cash Flow는 문자 그대로 자금의 흐름을 의미한다. 자금의 유입을 플러스, 지출은 마이너스로 표시하여 합계한 것이 Cash Flow이다. 따라서 판매대금의 입금은 플러스이고 수익지출(비용으로 지출되어 비용지출이라고도 한다)과 자본지출(자사의 증가를 수반하는 지출로 Plant 건설비 등이 포함)은 마이너스 요인이다. Cash Flow를 구하는 방법은 그림 4.2와 같다.

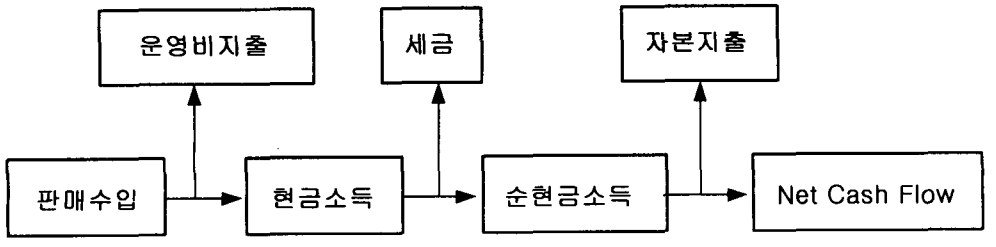


그림 4.2 Cash Flow 구하는 방법(1)

그림 4.2에서 운영비지출에는 감가상각비와 차입금의 이자는 포함

되지 않는다. 감가상각비는 현금지출을 수반하지 않는 비용이므로, 현금수익지출에는 포함되지 않는 것이 명백하지만, 이자까지도 포함하지 않는 이유를 다음과 같이 설명할 수 있다. 회계상의 계산에서는 차입금에 대해서 이자를 지불하는 것은 당연하며, 이자는 비용으로서 취급된다. 한편, 자기자금에 대해서는 같은 이유로 비용이 발생하지 않는 것으로 보여진다. 이러한 처리는 회계상으로는 올바를지 모르지만, 장래의 투자에 대한 경제성을 평가하는 경우에는 올바르지 않다. 왜냐하면, 자기자본을 타인에게 대부하면 당연히 이자를 받는 것이므로 자기 프로젝트에 대해서도 이자의 개념을 도입할 필요가 있다. 한편, Cash Flow의 목적은 복리공식을 사용하여 장래의 Cash Flow를 현재의 현가로 환산하는 것이고, 이 환산 때의 Cash Flow 전체에 대한 이자가 고려되어야 하므로 Cash Flow의 계산에서는 운영비 중에 이자를 포함시키지 않는다.

다음으로 세금은 과세대상소득(납세전 이익)×세율의 합산으로 구하는데 이 과세대상소득은 그림 4.2에서의 현금소득과 같지는 않다. 위의 설명에 의해서 현금수익지출에는 감가상각비와 이자가 포함되어 있지 않으므로,

$$\text{과세대상소득(납세전이익)} = \text{현금소득} + \text{감가상각비} + \text{이자} \quad (4.12)$$

이다.

이상과 같이 납세전 이익과 현금소득과의 관계가 명확해졌으므로, Cash Flow는 그림 4.3과 같이 두 가지 방법으로 계산할 수 있다. 이에 의하면 Cash Flow는 다음 식으로 표시된다.

$$\textit{Net Cash Flow} = \text{납세후이익} + \text{감가상각비} + \text{이자} - \text{자본적지출} \quad (4.13)$$

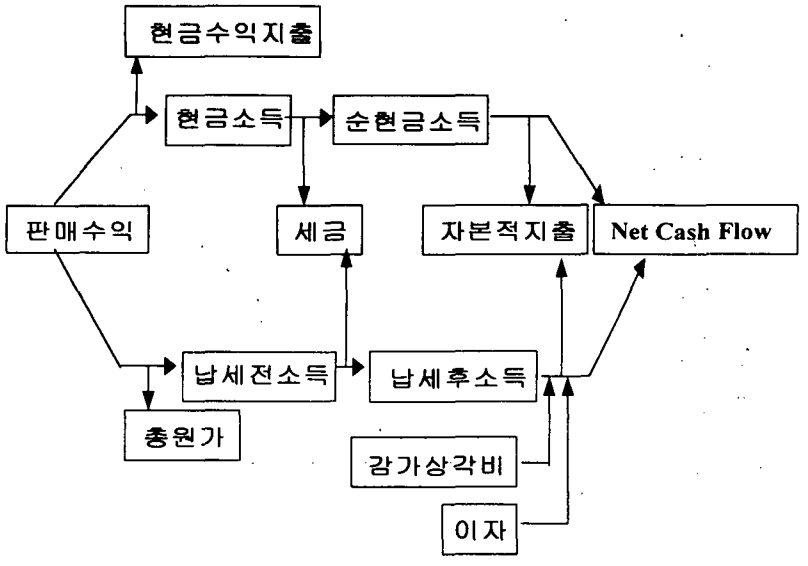


그림 4.3 Cash Flow 구하는 방법(2)

다음에 Cash Flow의 계산 예를 표 4.3에 표시한다. 이 예에 의해서 제14년째 자본지출이 플러스로 나타나 있는 것은 운전자금 10억원을 회수한 것을 의미한다. 또 이 예에서 누적 Cash Flow를 그래프로 나타내면 그림 4.4와 같다.

표 4.3 Cash Flow 계산 예

년	납세후순이익 (ANNP)	감가상각비 (AD)	이자 (AIN)	자본적지출 (ATC)	Cash Flow (ACF)	누적 Cash Flow (ΣACF)
0				- 100	-100	-100
1				- 300	-300	-400
2				- 600	-600	-1,000
3				- 7,500	-7,500	-8,500
4				- 1,500	-1,500	-10,000
5	1,200	900	600	0	+2,700	-7,300
6	1,600	900	540	0	+3,040	-4,260
7	2,000	900	480	0	+3,380	-880
8	2,000	900	420	0	+3,320	+2,440
9	2,000	900	360	0	+3,260	+5,700
10	2,000	900	300	0	+3,200	+8,900
11	1,700	900	240	0	+2,840	+11,740
12	1,400	900	180	0	+2,480	+14,220
13	1,400	900	120	0	+2,420	+16,640
14	1,200	900	60	+1,000	+3,160	+19,800

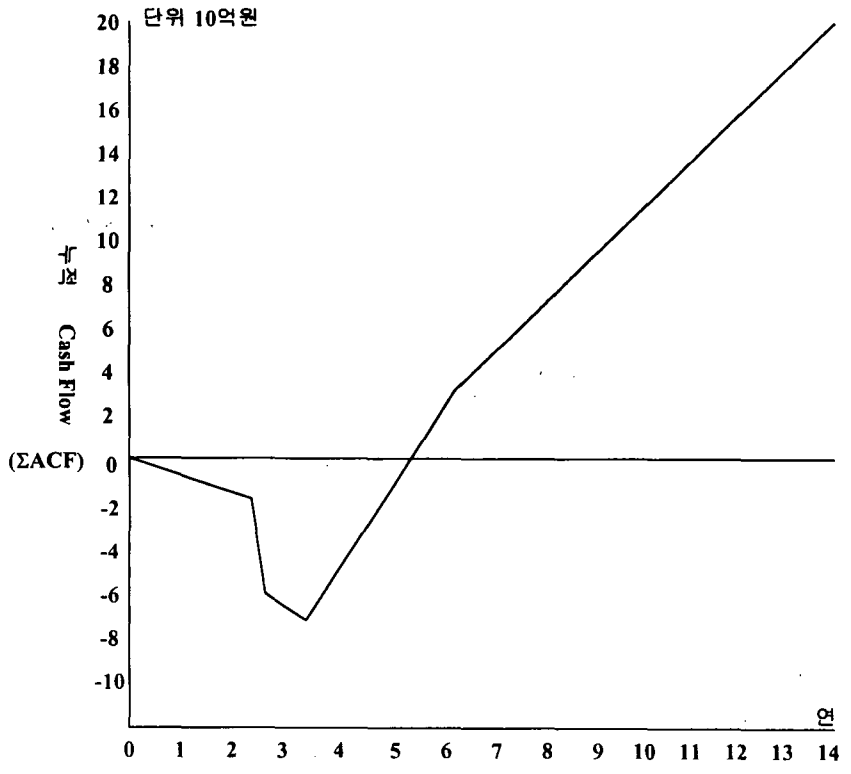


그림 4.4 누적 Cash Flow 곡선

3. DCF의 개요

앞에서 설명한 것과 같이 Cash Flow는 각 년도에 있어서 자금의 유입량을 단순히 계산한 것이지만, 경제성 평가를 위해서는 이 각년도의 Cash Flow를 복리공식을 사용하여 현가로 환산하여야 한다. 그러나, 반드시 현가로만은 아니고 종가로 환산해도 경제성 평가는 가능하지만, 우리들에게는 장래의 가치보다도 현재의 가치가 몸 가까이 느껴져 감각적으로 이해하기 쉬우므로 현가로 환산하는 것이 일반적이다. 장래의 어떤 연도에 있어서의 Cash Flow를 현가로 환산하는 것을 할인(Discount)라고 하고, 할인된 Cash Flow를 Discounted Cash Flow(DCF)라 한다. 즉, DCF는 다음

식과 같이 표시된다.

$$DCF = ACF(P/F, i, n) = \frac{ACF}{(1+i)^n} \tag{4.14}$$

상기 식에서 i는 할인율이고 주로 자금 비용을 연이율로 표시한 것으로서 경제성 평가에서 할인율로서 어떠한 이자율을 채용할 것인가 하는 것에 대해서는 엄밀하게는 여러 가지 논란의 여지가 있지만, 보통은 차입금에 대한 시장 이자율을 사용한다. 일반적으로 필요수익률(Minimum Attractive Rate of Return; MARR)이라고 명명되며, MARR(또는 최저 필요수익률)이란 투자자가 수용할 수 있는 투자안에 대한 수익률의 최저치를 말하며, 여기서는 MARR을 10%로 설정한다. 표 4.4는 전항에서 계산한 Cash Flow로부터 할인율 10 %를 사용하여 DCF를 구한 것이다. 또, 표 4.4에 근거하여 누적 DCF 곡선을 표시한 것이 그림 4.5이다. 그런데, 이 도표에서 i=0 % 곡선은 그림 4.4의 누적 Cash Flow 곡선과 동일하다.

표 4.4 DCF 계산 예

년	net cash flow (ACF)	누적 cash flow ($\sum ACF$)	할인율 10%		
			할인계수 (P/F,i,n)	DCF	누적 DCF
0	-100	-100	1.0000	-100	-100
1	-300	-400	0.9091	-273	-373
2	-600	-1,000	0.8264	-496	-869
3	-7,500	-8,500	0.7513	-5,635	-6,504
4	-1,500	-10,000	0.6836	-1,025	-7,529
5	+2,700	-7,300	0.6209	+1,676	-5,853
6	+3,040	-4,260	0.5645	+1,716	-4,137
7	+3,380	-880	0.5132	+1,735	-2,402
8	+3,320	+2,440	0.4665	+1,549	-853
9	+3,260	+5,700	0.4241	+1,383	+530
10	+3,200	+8,900	0.3855	+1,234	+1,764
11	+2,840	+11,740	0.3505	+995	+2,759
12	+2,480	+14,220	0.3186	+790	+3,549
13	+2,420	+16,640	0.2897	+701	+4,250
14	+3,160	+19,800	0.2638	+832	+5,082

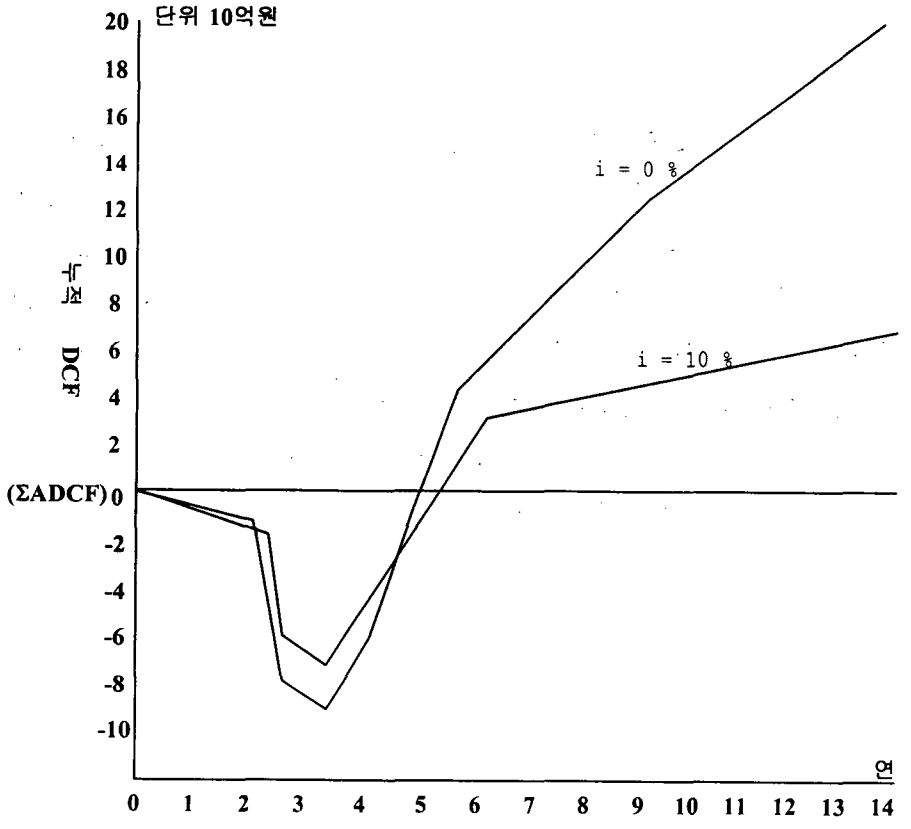


그림 4.5 누적 DCF 곡선

4. DCF에 의한 경제성 평가기준

다음으로 DCF에 의한 경제성 평가에서의 구체적인 평가기준에 대해서 알아본다. 여기서 주의할 것은 하나의 기준만으로 완전한 평가를 할 수 없다는 것이다. 어떤 기준을 판단의 재료로 삼느냐에 따라 때로는 완전히 역의 결과를 가져온다. 최종적으로 어떤 투자안을 선택하는가 하는 것은 의사결정의 문제이며, 다음의 각종 경제성 평가기준은 의사결정에 필요한 자료인 것이다. 또, DCF 자체가 매상고, 비용 등에 대한 예상치를 기초로 계산된 것이므로 DCF에만 근거하여서는 경제성 평가기준의 정도(精度)를 상회 할 수는 없다. 즉, 불확실성 하에서의 기준이므로, 그 한계에 대해 충분히 이해한 다음에 활용해야만 한다. 다음 각종의 경제성 평가기준을 시간, 자본, 이자의 인자로 분류하여 알아본다.

(1) 시간인자

① Project의 완료기간

이것은 Project의 출발부터 공장을 폐기할 때까지의 기간을 가리키며, 이기간은 평가기준으로는 별로 의미를 가지지 않지만, 경제성 계산의 전제 조건으로서는 중요하다. 이기간을 도시하면 그림 4.6과 같다.

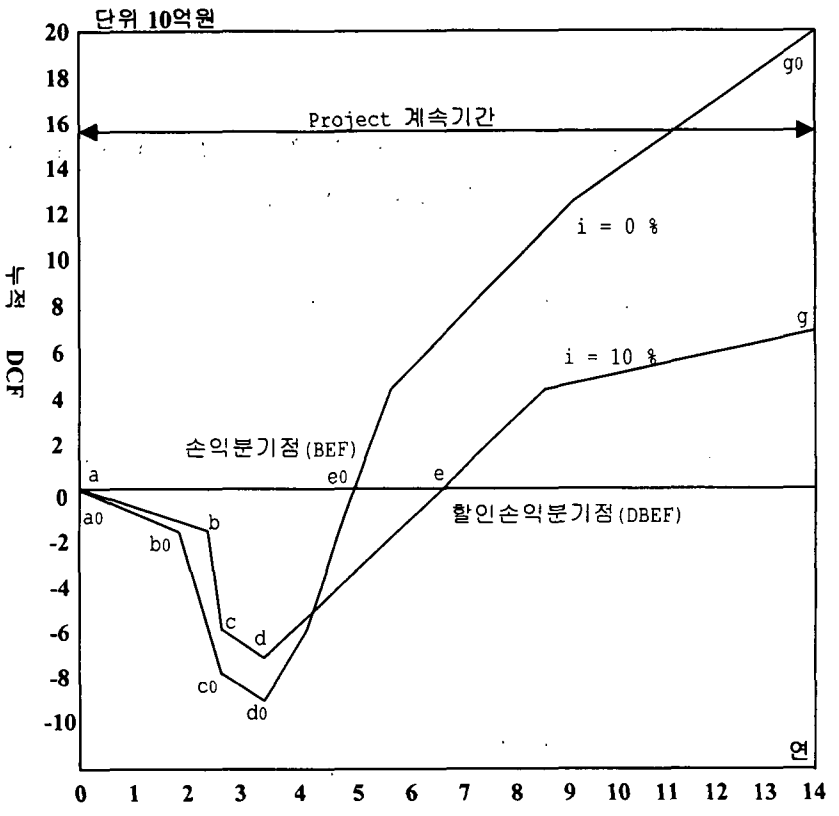


그림 4.6 Project 계속기간과 손익분기점

② 손익분기점(Break Even Point, BEP)

과거에 보통 사용된 손익분기점이란 용어는 수입과 지출이 같게되는 것을 의미했다. 즉 Production Rate 손익분기점이라고 부를 수 있는 것이었다. 여기서 사용하고 있는 손익분기점은 투자한 자금이 전액 회수된 시점을 Project 개시 때부터의 기간(년도)으로 표시한 것이다.

손익분기점은 그림 4.6과 같이 누적 CF 선도(즉 i=0 %의 경우)과 누적 DCF 곡선의 두 개로 나타내고, 전자의 경우를 손익분기점(BEP), 후자의 경우를 할인 손익분기점(Discounted Break Even Point, DBEP)이라고

부른다. 손익분기점의 연수가 짧은 쪽이 좋은 것은 물론이다.

③ 최대 투자상등기간 (Equivalent Maximum Investment Period, EMIP)

Project의 투자는 전액이 일시에 행해지지 않고 수년간에 걸쳐서 분할투자가 행해진다. 또 자금의 회수도 한번으로 그치지 않고 매년 매상 수익이 누적되어 어떤 시기(위에서는 BEP시)에서 전액을 회수하게 되는 것이다. 여기서 만약 자금이 단번에 전액 투자되어 또 단번에 전액 회수된다고 가정하면 이 투자로부터 회수까지의 기간을 최대 투자상등기간(EMIP)이라 하고 다음과 같이 정의된다(그림 4.7 참조).

$$EMIP = \frac{\text{면적}(a_0 - e_0)}{-ACF\max} \tag{4.15}$$

즉, $EMIP \times (-ACF\max) = \text{면적}(a_0 - e_0)$이다. 따라서 그림 4.7에서 면적 면적$(a_0 - e_0)$와 같은 사각형을 그림의 점선과 같이 만들 때 EMIP는 그 사각형의 가로길이로 표시된다.

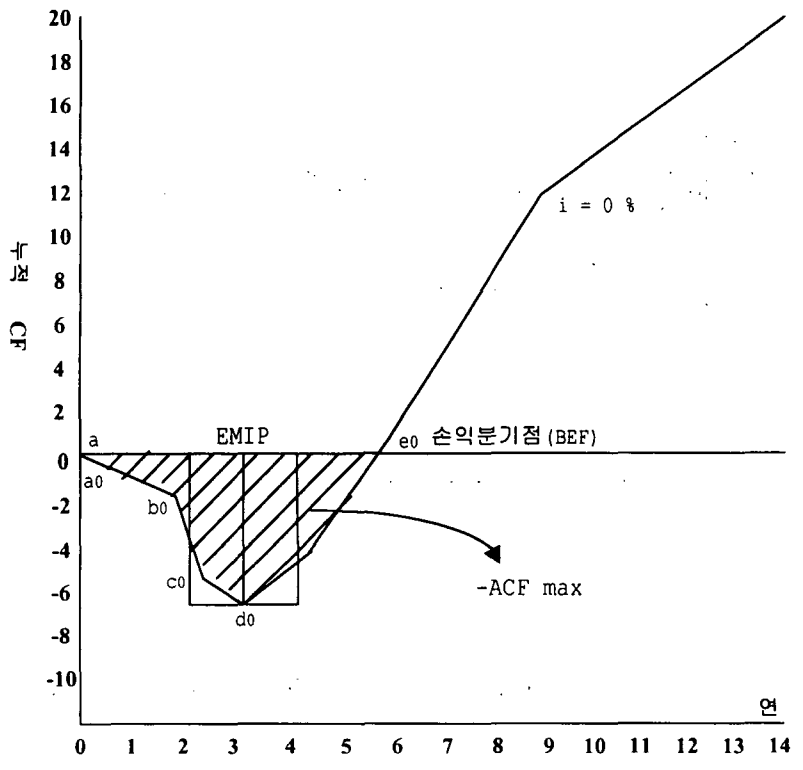


그림 4.7 최대투자상등기간(EMIP)

앞에서 설명한 손익분기점(BEP)는 투자개시 시점으로부터 회수종료시점까지의 기간을 나타내지만, 이 기간만으로는 투자로부터 회수까지의 경향은 알 수 없다. 즉, 그림 4.8과 같이 손익 분기점이 같아도 형상이 다른 DCF곡선이 몇 개라도 생길 수 있다. 투자 자본은 손익분기점까지에는 회수 될 수 있을지 여부의 위험을 안고 있지만, 이 위험은 예를 들어 손익분기점이 같아도 DCF 곡선의 형상에 따라 달라지게 된다. 따라서 손익분기점의 불충분한 바를 보완하기 위해 최대 투자상등기간(EMIP)의 개념이 도입된 것이다. 다른 조건이 같다면 최대 투자상등기간이 짧은 쪽이 경제성이 높다. 그러나 이 EMIP는 보통 누적 CF 즉 할인율 i=0%의 경우에 대해서만 구하면 무방하다.

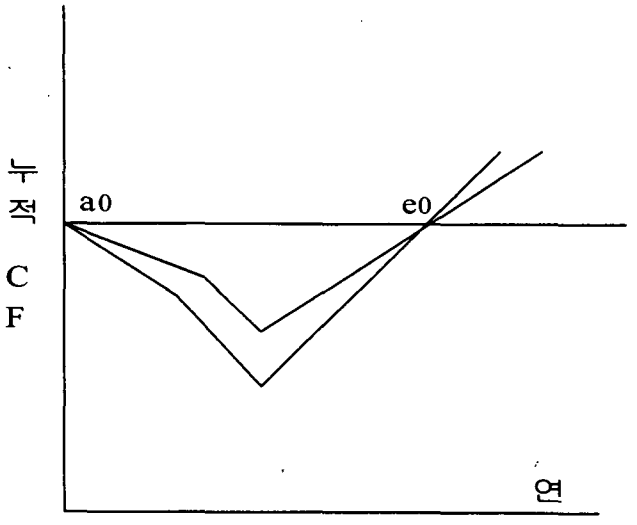


그림 4.8 손익분기점이 동일한 2 개의 Project

④ 이자회수기간(Interest Recovery Period, IRP)

투자의 목적은 투자액 이상의 수익을 얻는 것이므로 손익분기점에서 투자액을 전액 회수하는 것만으로는 만족할 수 없다. 따라서 손익분기점을 구할 뿐 아니라, 손익분기점 이후의 DCF 곡선의 경향도 검토해야 한다.

이 경향을 측정하는 지표가 이자회수기간(IRP)이다. 그림 4.9에서 횡축을 기준으로 아랫부분의 면적과 윗부분의 면적을 같게 하는 f점을 구하고, Project 개시 때부터 f점까지의 기간(년도)을 이자 회수기간이라고 정의한다.

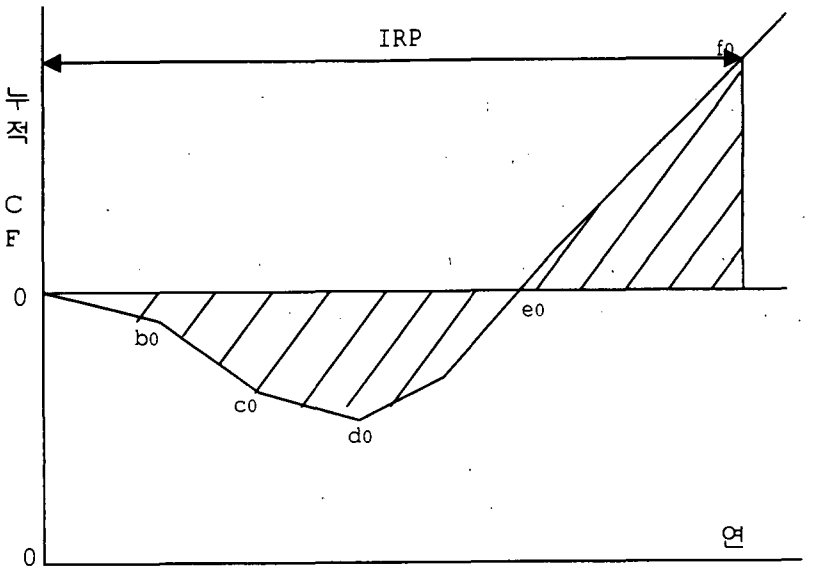


그림 4.9 이자 회수 기간(IRP)

이자 회수기간이 의미하는 것은 다음과 같다. Project가 개시한 때부터 손익분기점까지는 항상 자금이 지출초과의 상태이므로 자금 Cost, 즉 지출이자가 발생한다고 볼 수 있다. 한편, 손익분기점 이후는 자금의 수취초과이고 이의 누적액은 매년 증가하는데 이 누적자금에 대해서도 이자가 발생하고 있다고 생각하여, 결국 현금소득으로부터 일부러 이자를 현금으로 지출하지 않고도 자금의 보유로 어떤 시기에는 손익분기점까지의 이자와 상쇄될 수 있는 이자가 생긴다는 계산이다. 이자 회수기간이란 것은 이상과 같은 사고방식 하에서 걸려있는 이자를 회수할 수 있는 기간을 나타내고 있다. 이 기간도 손익분기점 및 최대 투자상등기간과 같이 짧은 것이 좋다.

⑤Payback 기간 (Payback Period, PBP: 자본회수기간)

Payback 기간은 누적 Cash Flow(CF)에서 고정 자산지출액을 회수할 때까지의 연수를 표시한다. 고정자산 중에는 상각하지 않는 것(예를 들어 토지 등)도 있는데 여기서 말하는 고정자산이란 상각하는 것만을 가리키고 잔재가치는 포함하지 않는다. 각 년도에서의 Cash Flow를 ACF, 고정자산총액을 CFC, 잔재가치를 S로 표시하면 Payback 기간(PBP)는 다음 식을 만족하는 n(년도)로서 구해진다.

$$\sum_{n=0}^{n=PBP} ACF = CFC - S \qquad (4.16)$$

Payback 기간도 짧은 것이 좋지만, 이 지표만으로 판단한다면 틀린 결과를 얻게 된다. 예를 들어 표 4.5와 같은 Cash Flow를 가진 2개의 Project A, B를 비교해 보자. 이 예에서는 Payback 기간(PBP)은 Project A편이 짧지만, 수익성은 Project B편이 좋다.

표 4.5 Payback 계산 예

연	Project A의 Cash Flow	Project B의 Cash Flow
0	-1,000 백만원	-1,000 백만원
1	500	0
2	300	100
3	200	200
4	100	300
5	0	400
6	0	500
7	0	600
PBP	3년	5년

(2) 자본인자

① 최대 누적지출액(Maximum Cumulative Expenditure)

Project에서 자본지출 누적액의 최대치이며, $-\sum ACF\max$ 라고 표시된다. 여기서 ACF는 할인하지 않는 년간 Cash Flow의 금액을 의미한다. 마이너스가 붙은 것은 Cash Flow 중에서 자본지출은 마이너스로서 계산되어 자본 지출액으로서의 절대치를 취하기 때문이다. 그림 4.10에서 최대 누적지출액은 점 d_0의 금액의 절대치로서 구해진다.

② 최대 할인 누적 지출액(Maximum Discounted Cumulative Expenditure)

①항의 최대 할인누적 지출액은 할인하지 않은 누적 Cash Flow에 근거한 최대 지출액을 가리키는데 대해서, 본 항의 최대 할인 누적 지출액은 누적 DCF에 근거한 것이고, $-\sum ADCF$ max로 표시된다. 그림 4.10에서 이것은 점d로 표시되어 점d_0와의 관계는 명확하다. 또 $-\sum ACF$ max가 $-\sum ADCF$ max 보다 큰 것도 그림 4.10으로부터 이해될 수 있다.

③ 누적 Cash Flow (Cumulative Net Cash Flow)

이것은 Cash Flow의 Project 최종 년도에서의 누계액이며 $-\sum ACF$로 표시된다. 그림 4.10에서 g_0 점의 크기이다. 이 지표는 할인하지 않은 Cash Flow로부터 구해져 이자를 전혀 고려하지 않은 때문에 경제성 평가에서는 다음의 순현가(NPV)쪽이 훨씬 중요한 지표이다.

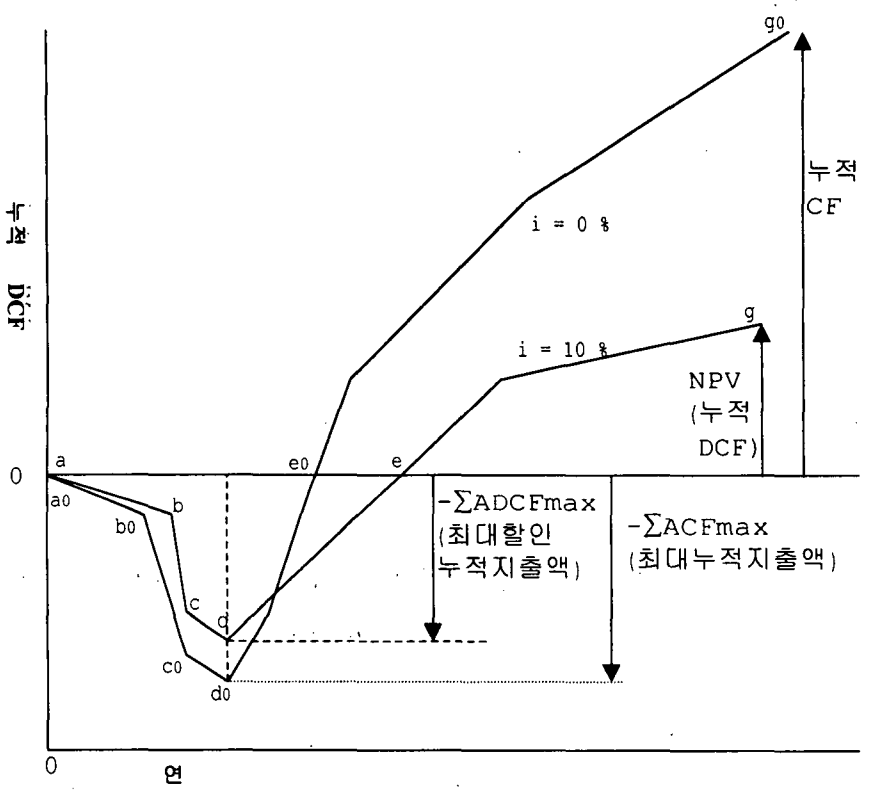


그림 4.10 자본인자 관련 설명

④ 누적 DCF(Cumulative Net Discounted Cash Flow) 또는 순현가(NPV)

각 년도의 Cash Flow를 적당한 이자율로 할인한 뒤에 누적한 것이다. 따라서 이 수치는 채용한 이자율에 따라 다르다. 할인율은 특별히 결정되어 있지는 않지만, 시장 이자율이라고 생각되는 10%를 많이 사용한다. 이 NPV는 투자의 가치를 직접적으로 표시하는 것이어서, 여기서 설명하는 수많은 경제성 평가지표 중에서 수익률(IRR)과 더불어 가장 중요한 의미를 지니는 것이다. 제n년도의 Cash Flow를 ACF_n 이라고 표시하면 NPV는 다음식으로 구해진다.

$$NPV = \sum_{o}^{n} ACF_n \cdot (P/F, i, n) = \sum_{o}^{n} ACF_n \cdot \frac{1}{(1+i)^n} \quad (4.17)$$

윗 식에 따라 실제로 NPV를 구하는 대는 표 4.6의 순서에 따라 하는 것이 좋다.

표 4.6 NPV 계산순서

연	Cash Flow	할인계수	DCF
0	ACF_0	1	ACF_0x1
1	ACF_1	$1/(1+i)$	$ACF_1/(1+i)$
2	ACF_2	$1/(1+i)^2$	$ACF_2/(1+i)^2$
·	·	·	·
·	·	·	·
·	·	·	·
n	ACF_n	$1/(1+i)^n$	$ACF_n/(1+i)^n$
	합계=누적CF		합계=NPV

⑤ 자본화 비용(Capitalized Cost)

설비는 어떠한 것도 영구히 사용될 수 없고 일정한 수명이 있다. 따라서 사용할 수 없게된 때에는 신품과 교환하지 않을 수 없다. 설비에는 각각 고유의 수명이 있으므로 경제성을 평가하는데는 표면적인 가격 뿐만 아니라 그 수명도 고려되어야 한다. 예를 들어 설비 A는 1억원으로 내용

년수가 7년인데 반해, 설비 B는 1.2억원으로 내용 년수가 10년이라면 어떤 설비가 경제적인가 하는 문제가 생긴다. 이 문제를 해결하기 위해 도입된 것이 자본화 Cost 라는 개념이다. 자본화 Cost는 어떤 설비를 내용 년수에 관계없이 영구히 사용한다고 가정한 경우의 원가를 의미한다. 자본화 q 비용을 CK, 설비비(Fixed Capitalized Cost)를 CFC, 내용 년수를 n년, n년 후의 설비잔존가치(Salvage or Scrap Value)를 S, 연이율을 i라 하면 다음 관계식이 성립된다.

$$(CK-CFC)(1+i)^n = CK-S \tag{4.18}$$

윗 식을 변형해서,

$$CK=[CFC-\frac{S}{(1+i)^n}][\frac{i(1+i)^n}{(1+i)^n-1}]\cdot\frac{1}{i} \tag{4.19}$$

또는,

$$CK=(CFC-S(P/F,i,n))(A/P,i,n)\cdot\frac{1}{i} \tag{4.20}$$

$$(P/F,i,n)=\frac{1}{(1+i)^n} \quad \text{(현가 계수)}$$

$$(A/P,i,n)=\frac{(1+i)^n}{(1+i)^n-1} \quad \text{(자본화 비용계수)}$$

(3) 이자인자(利子因子)

① 수익률(Internal Rate of Return(IRR), Discounted Cash Flow Rate of Return(DCFRR))

투자의 가치를 현가로 나타낸 것이 NPV인데, 이 NPV는 할인율에 따라 변동한다. 그림 4.11을 보면 할인율이 크면 NPV는 작아진다.

또 NPV를 0으로 되게 할인율을 정할 수도 있다. 이렇게 결정된 할인율을 수익률이라 한다. 즉,

$$NPV = \sum ADCF$$
$$= ACF_0 + \frac{ACF_1}{1+i} + \frac{ACF_2}{(1+i)^2} + \cdots + \frac{ACF_n}{(1+i)^n} = 0 \qquad (4.21)$$

를 만족시키는 i(i^*)가 수익률이다.

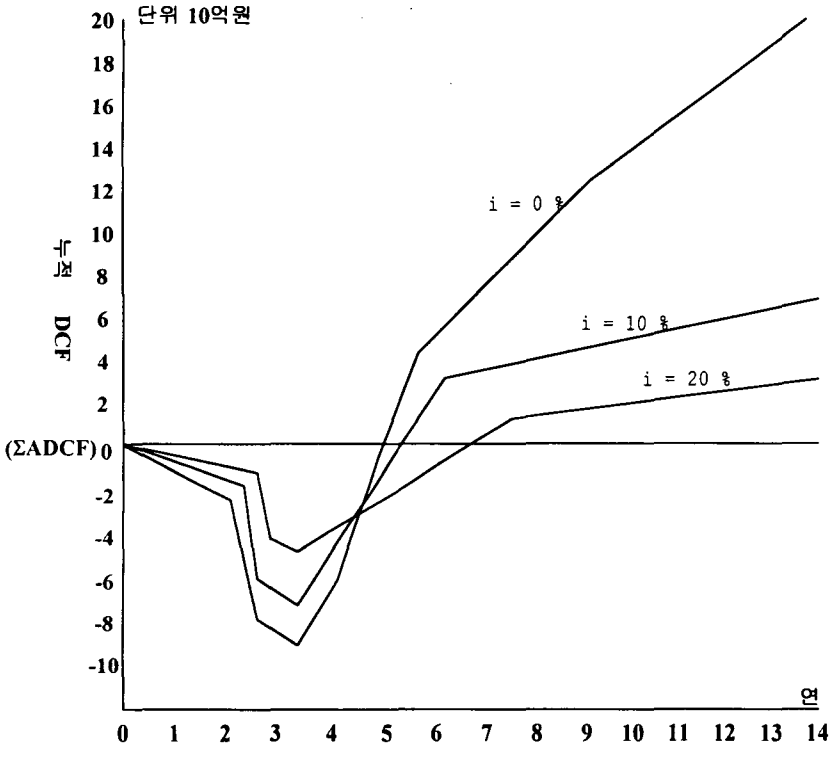


그림 4.11 DCF 투자 이익율

순현가(NPV)를 구할 때에는 i로서 MARR(예를 들면 연 10 %)를

사용한다. 이 의미는 투자자가 수락할 수 있는 최소한의 시장 이자율로써 자금을 운용하여 투자안을 실행할 경우 순현가(NPV)와 같은 가치의 현금 소득이 기대 될 수 있다는 것이다. 따라서 예를 들어 순현가(NPV)를 0으로 하는 할인율을 15 %라 하면, 시장 이자율이 15 % 또는 그 이상일 때는 투자의 이익이 전혀 없고, 시장 이자율이 10 %일 때는 15 %와 10 %의 차이가 실질적인 이익률을 나타낸다.

수익률을 식(4.21)에 의해 구하면 n차 방정식이 되는데 n가 4까지의 경우 외에는 대수적으로 해를 구할 수 없다. 따라서 i에 적당한 이자율을 주어 NPV를 0으로 만드는 i를 구할 수 밖에 없다. 그러나 실제로는 NPV를 꼭 0으로 만드는 i를 2개 구하여 (보통 NPV가 0을 기준으로 +와 -로 되는 2개의 i를 구한다), 보간법에 의해 근사적으로 i를 계산한다.

표 4.7(a) DCF 투자 이익율 계산표

년	Cash Flow (ACF)	누적 Cash Flow ($\sum ACF$)	할인율 10%		
			현가계수 (fa)	DCF (ADCF)	누적 DCF ($\sum ADCF$)
0	-100	-100	1.0000	-100	-100
1	-300	-400	0.9091	-273	-373
2	-600	-1,000	0.8264	-496	-869
3	-7,500	-8,500	0.7513	-5,635	-6,504
4	-1,500	-10,000	0.6836	-1,025	-7,529
5	+2,700	-7,300	0.6209	+1,676	-5,853
6	+3,040	-4,260	0.5645	+1,716	-4,137
7	+3,380	-880	0.5132	+1,735	-2,402
8	+3,320	+2,440	0.4665	+1,549	-853
9	+3,260	+5,700	0.4241	+1,383	+530
10	+3,200	+8,900	0.3855	+1,234	+1,764
11	+2,840	+11,740	0.3505	+995	+2,759
12	+2,480	+14,220	0.3186	+790	+3,549
13	+2,420	+16,640	0.2897	+701	+4,250
14	+3,160	+19,800	0.2638	+832	+5,082

표 4.7 DCF 투자 이익율 계산표

년	Cash Flow	누적 Cash Flow	할인율 10%			할인율 20%			할인율 23%		
			현가 계수	DCF	누적 DCF	현가 계수	DCF	누적 DCF	현가 계수	DCF	누적 DCF
	(ACF)	ΣACF	(P/F,i,n)	(ADCF)	ΣADCF	(P/F,i,n)	(ADCF)	ΣADCF	(P/F,i,n)	(ADCF)	ΣADCF
0	-100	-100	1.0000	-100	-100	1.0000	-100	-100	1.0000	-100	-100
1	-300	-400	0.9091	-272	-373	0.8333	-250	-350	0.8130	-244	-344
2	-600	-1,000	8.8264	-496	-869	0.6944	-417	-767	0.6610	-397	-741
3	-7,500	-8,500	0.7513	-5,635	-6,504	0.5787	-4,340	-5,107	0.5374	-4,031	-4,772
4	-1,500	-10,000	0.6836	-1,025	-7,529	0.4823	-723	-5,830	0.4369	-655	-5,427
5	+2,700	-7,300	0.6209	+1,676	-5,853	0.4019	+1,085	-4,745	0.3552	+959	-4,468
6	+3,040	-4,260	0.5645	+1,716	-4,137	0.3349	+1,018	-3,727	0.2888	+878	-3,590
7	+3,380	-880	0.5132	+1,735	-2,402	0.2791	+943	-2,784	0.2348	+794	-2,796
8	+3,320	+2,440	0.4665	+1,549	-853	0.2326	+772	-2,012	0.1909	+634	-2,162
9	+3,260	+5,700	0.4241	+1,383	+530	0.1938	+632	-1,380	0.1552	+506	-1,656
10	+3,200	+8,900	0.3855	+1,234	+1,764	0.1615	+517	-863	0.1262	+404	-1,252
11	+2,840	+11,740	0.3505	+995	+2,759	0.1346	+382	-481	0.1026	+291	-961
12	+2,480	+14,220	0.3186	+790	+3,549	0.1122	+278	-203	0.0834	+207	-754
13	+2,420	+16,640	0.2897	+701	+4,250	0.0935	+226	+23	0.0678	+164	-590
14	+3,160	+19,800	0.2633	+832	+5,082	0.0779	+246	+269	0.0551	+174	-416

그러면 표 4.7을 보며 DCF 투자 이익율을 구해본다. 표에 의하면 할인율이 20%이면 누적 DCF (NPV)는 +269, 할인율이 23%이면 -146이므로 NPV가 0이되는 할인율은 20%와 23%사이에 있다. 할인율과 NPV와의 관계는 근사적으로 그림 4.12과 같이 나타나므로 다음 관계식이 성립한다.

$$269 : x = 416 : (3 - x)$$

이것을 풀면

$$416x = 269(3 - x)$$
$$x = 1.2$$

따라서 NPV를 0으로 하는 할인율은 다음과 같다.

$$IRR = 20\% + x\%$$
$$= 21.1\%$$

그런데 수익률은 앞서 설명한 NPV와 더불어 경제성 평가에서 가장 중요한 지표이다.

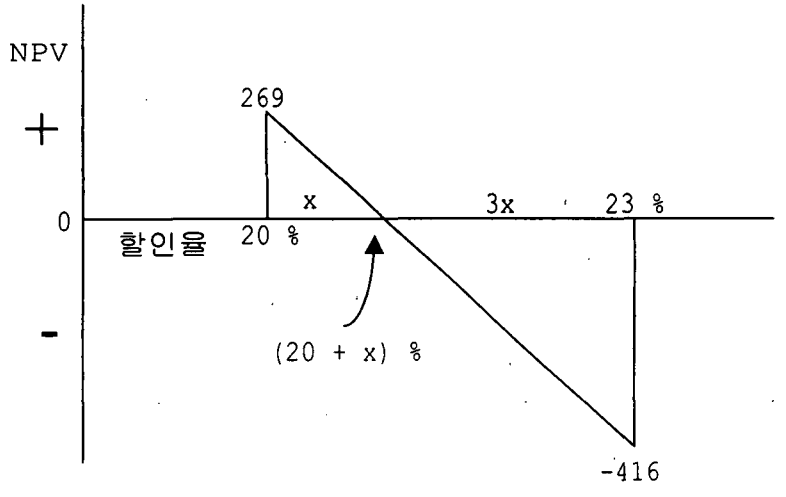


그림 4.12 투자 이익률의 보간법 계산

② 자본 이익비(Capital Rate of Return Ratio, CRR)

앞서의 NPV는 투자의 가치 또는 이익을 나타내는 중요한 지표이다. 각종 투자안의 이익의 대소는 NPV에 의해서 바로 판정된다. 그러나 투자액에 대한 비율로써 수익성을 알아볼 때는 NPV만으로는 불충분하다. 이런 의미에서 NPV를 보충하기 위해서 도입된 자료가 자본 이익비(CRR)이다. 자본 이익비은 NPV를 자본지출 총액으로 나눈 것이다.

$$\text{자본이익비}(CRR) = \frac{\text{순현가}}{\text{자본지출총액}} = \frac{NPV}{-\sum ACF\max} \qquad (4.22)$$

윗 식에서 $-\sum ACF\max$는 누적 Cash Flow 곡선의 최하점에서의 금액의 절대치 나타낸다. NPV가 같은 2개의 Project가 있다면 큰 쪽이 효율이 좋은 투자다.

이상으로 경제성 평가의 지표에 대한 정의와 의미를 설명했다. 위에서 설명한 지표 중에서 가장 중요한 것은 NPV와 IRR이다. 경제성이 없다는 것은 NPV가 마이너스이든가 IRR<MARR 인 경우이다. NPV가 마이너스이라는 것은 투자액을 전액회수 할 수 없다는 것을 의미하고, 또 NPV는 MARR로써 할인한 수치이므로 NPV가 마이너스인 것은 IRR이 MARR보다 작다는 것과 같다.

IRR이 MARR보다 작으면, 투자자금을 은행으로부터 차입한 경우에서 수익만으로는 갚을 수 없다는 것을 의미하고 또 자기자금을 투자한 경우에는 투자로부터의 그 수익이 그 자금을 직접 타인에게 임대할 때 받을 수 있는 이자보다 작다는 것을 의미한다.

그러나, 경제성 평가는 반드시 단순한 것은 아니다. 만약 단일 지표로 평가한다면 답은 Yes 나 No로 판명되지만 역으로 그 간단함으로 인하여 틀린 답이 생길 가능성이 매우 크다. 예를 들어 위의 NPV와 IRR이 가장 중요한 지표라 해서, 이 2개의 지표만으로 판정하면 위험하다. Project A, B에서 NPV로부터 보면 A가 좋고, IRR로 보면 B가 좋은 경우에는 어떤 판정을 하면 좋을까? 어느 한쪽이 일방적으로 결정할 수 없다. 이런 경우에는 다시 다른 어떤 관점으로 검토해야 한다. 여기서 소개할 각종의 지표는 편협한 판정을 피하기 위해서 여러 가지 다른 관점에 근거하여 생긴 것들이다. 따라서 판정의 실수에 대한 가능성을 줄였지만, 지표가

많기 때문에 판정이 단순하지는 않다. 지표에 따라서는 상반되는 평가가 생기는 경우도 많은데 모든 지표로부터 어떠한 종합적인 판정을 내릴 것인가 하는 것은 기업의 Management의 의지결정 문제이지만 자본인자로는 NPV, 이자인자로는 수익률 그리고 보조적 측도인 payback 기간이 주로 사용되므로 본 과제의 경제성 평가측도는 이를 중심으로 계산한다.

Project의 경제성 평가는 복수의 계획을 선정하여 이들에 대해서 경제성 평가를 실시하고 상대적으로 유리한 계획을 채용하는 순서를 밟는 것이 보통이다. Project 에 대한 경제성 평가지표를 표 4.8과 같이 각 지표에 대한 특성을 간단히 설명한다.

표 4.8 경제성 평가지표 비교

경제성 평가지표	비교
1. 시간 인자	
계속 기간	
손익 분기점	작을수록 좋다
할인율 0 % (BEP)	〃
할인율 10 % (DBEF)	〃
최대 투자상등기간 (EMIP)	〃
이자 회수기간 (IRP)	〃
Pay Back 기간 (PBP)	〃
2. 자본 인자	
최대 누계지출액	
할인율 0 % (-∑ACFmax)	작을수록 좋다
할인율 10 % (-∑ADCFmax)	〃
최대 누계회수액	
할인율 0 % (∑ACFmax)	클수록 좋다
할인율 10 % (∑ADCFmax, NPV)	〃
자본화 비용	작을수록 좋다
3. 이자 인자	
수익률 (IRR)	클수록 좋다
자본이익비 (CRR)	〃

5. 감도분석(Sensitivity Analysis)

경제성 평가에서 아무리 고도의 기술을 사용해도 그 정도는 기초로 한 Data의 정도를 상회할 수 없고, 또 경제성 평가의 기초 Data인 판매수입, 원료 및 그외 비용, 설비비 또 이들로부터 만든 Cash Flow 등도 모두 한정된 과거의 실적 및 불확실한 미래의 예측에 근거 한 것이다.

경제성에 영향을 미치는 주요 요인으로서 매상수입, 비용, 설비비 등이 있다. 이들의 변동이 NPV에 미치는 영향은 경제 전문가의 전문적인 영역에 속하고 이에 대한 해석은 복잡하고 본 연구의 범위에 벗어나므로 본 연구에서는 감도분석은 생략한다.

6. 경제성 평가의 순서

이상으로 고전적인 경세성 평가와 DCF에 근거한 새로운 경제성 평가에 대한 설명을 마치고 다음으로 이들을 정리하여 일반적인 경제성평가의 순서를 나타내보자. 일반적으로 Project의 경제성 평가는 다음과 같은 순서로 작업이 진행된다. 다만 현실적으로는 작업 중에 이때까지의 결과가 처음으로 Feed-Back되어 Project안이 도중에 수정되는 경우가 많다.

(1) Project안의 선정 (통상 복수안)
(2) 투자액의 견적
(3) 자금 조달원천의 결정
(4) 수익계산
(5) DCF계산
(6)경제성 평가지표

(7)감도분석

(8)의사결정(최종안의 결정)

(2)부터 (6)항 까지는 계산순서를 표 4.9, 표 4.10, 그림 4.13, 그림 4.14에 표시한다.

표 4.9 투자액의 견적

1. 상각고정자산비	(→감가상각비)
Plant 건설비	
창업 준비비	창업 준비비와 건설기간 이자는 엄밀하게는 고정자산은 아니지만, 보통 편리상 고정 자산이라고 생각하여 계산하고 있다.
건설기간이자	
2. 토지대	토지를 구입하는 경우에는 자본적 지출이 되므로 토지대는 투자액에 포함되지만 토지를 임차하는 경우에는 부동산 임차료는 수익적 지출이 되므로 제조원가에 포함된다.
3. 운전자금	운전자본은 일반적으로 자본적 지출이라고 간주하지만 전액을 차입금으로 조달하는 경우는 수익적 지출로 취급할 수 있다.
4. 총투자액(1+2+3)	

표 4.10 자금조달 원천의 결정

총투자액
1. 내부원천 사내유보 감가상각
2. 외부원천 자기자본 : 자본금(신주식발행) 타인자본 : 차입금 · 회사채(→이자발생)

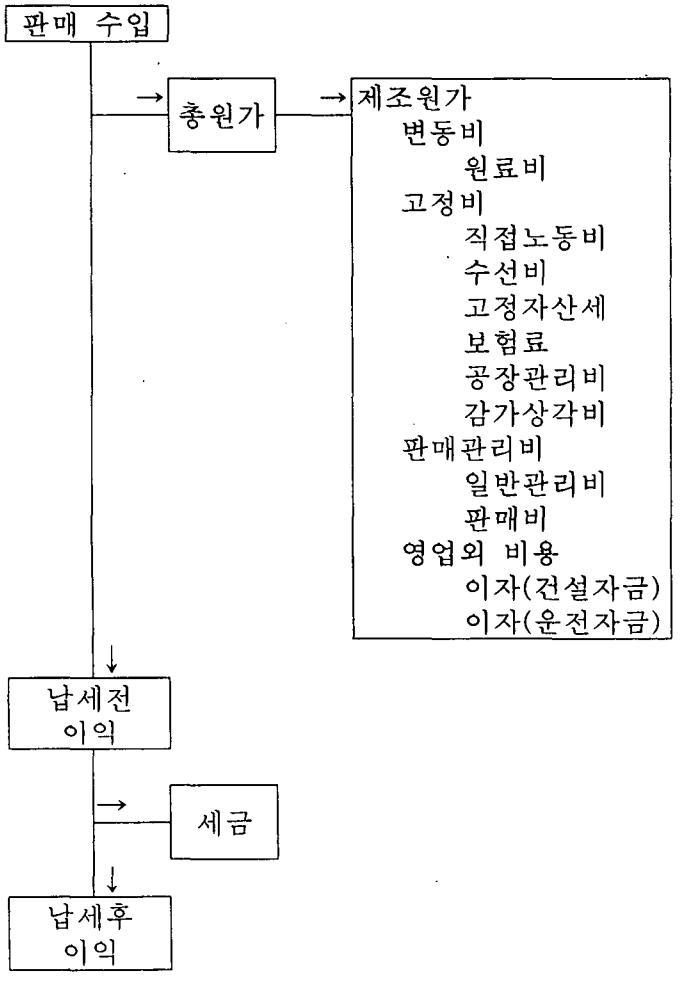
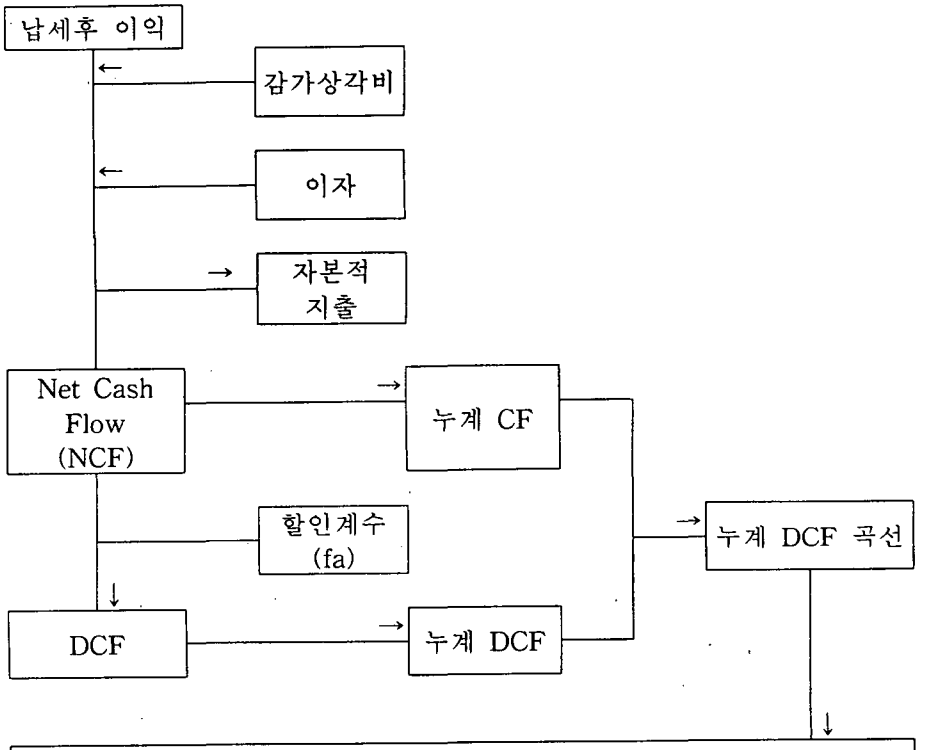


그림 4.13 수익계산



경제성 평가 지표	
시간 인자	계속기간 손익 분기점 : 할인율 0% (BEF) 손익 분기점 : 할인율 10% (DBEP) 최대투자 상등기간 (EMIP) 이자회수기간 (IRP) Payback 기간 (PBP)*
자본 인자	최대누계지출액 : 할인율 0% (-∑ACFmax) 최대누계지출액 : 할인율 10% (-∑ADCFmax) 최대누계회수액 : 할인율 0% (-∑ACFmax) 최대누계회수액 : 할인율 10% (-∑ADCFmax, NPV)* 자본화 비용 (CK)
이자 인자	수익률 (IRR)* 자본이익비 (CRR)

(주) * 일반화된 경제성 평가 지표

그림 4.14 DCF 계산과 경제성 평가지표

7. 개발 시스템에 대한 경제성 평가계산

지금까지 소개한 경제성 평가에 대한 이론적 배경을 실제 본 연구 개발 시스템을 현장에 적용할 경우에 대해 다음의 전제 조건하에서 계산하여 경제성 결과에 대해 알아본다.

(1) 연간 생산량

■ 수작업의 경우

- 1 Petri Dish = 6 개의 감자줄기(1 Node)를 투입
- 1인 1일 생산량 = 300 Petri Dish x 6 개 = 1,800 개
  (8시간/1일, 3.75개/1분)
- 불량율 10 %을 고려한 1인 1일 생산량 = 1,620 개
- 1인 연간 생산량 = 1,620 개 x 300 일 = 486,000 개
- 5인 연간 생산량 = 486,000 개 x 5 인 = 2,430,000 개

■ 자동화 시스템의 경우

- 1대/분 생산량 = 15 개
- 1대 1일 생산량 = 15 개 x 60 분 x 20 시간 = 18,000 개
- 불량율 5 %을 고려한 1대 1일 생산량 = 17,100 개
  (청정시설을 완비한 경우로 가정)
- 1대 연간 생산량 = 17,100 개 x 300 일 = 5,130,000 개

단, 1차 년도의 생산량은 상기량의 1/2로 간주함

(2) 고정 자산 투자비의 견적은 다음과 같다.

■ 공장구입대	0.5 억원
■ 시스템 구축비용	
• 수작업 경우 시스템구축비용	0.3 억원
• 자동화 경우 시스템 구축비용	2.0 억원

(창업 준비비 및 건설중 금리를 포함)

- 합 계 (총투자액)
- 수작업의 경우 총투자액 : 0.8 억원
- 자동화의 경우 총투자액 : 2.5 억원

(3) 총투자액 0.8 억원과 2.5 억원의 지출 스케쥴은 다음과 같다.

- 수작업 경우

  0연도 : 100 %
- 자동화 경우

  0 연도 : 40 %

  1 연도 : 60 %

(4) 운전자금은 매상고의 3개월 분으로 하고, 전액 은행으로부터 12%의 금리로 차입한 것으로 가정하며 수익적 지출로 간주한다.

(5) 총투자액 2.5 억원의 조달자원은 다음과 같다.

- 자본금 0.3 억원(수작업) 1.0 억원(자동화)
- 차입금 0.5 억원(수작업) 1.5 억원(자동화)

차입조건은 금리율 10 %, 원금을 5년 균등 상환하는 것으로 한다.

(6) 수익계산을 위한 줄기 1 Node 당 비용계산에 대해 다음과 같은 단가 기준으로 계산한다.

- 배양액 원가 1개당 : 2 원
- 세척비용 1개당 : 0.2 원
- 멸균비용 1개당 : 0.2 원
- 전기 비용 : 1 원
- 수도 비용 : 0.1 원

- 수작업의 경우 : 작업자 6명
- 자동화의 경우 : 작업자 1명

(7) 계획기간을 장비의 실질내용연수로 추정되는 10년으로 설정한다.

경제성 평가에 기초적으로 필요한 각 연도의 손익계산서(수작업의 경우 1~10년, 자동화 경우는 2~10년)를 표 4.11에 정리하였으며 자동화 경우 1차년도 손익계산서는 연간 생산량과 변동비(판매비 포함)만 2차년도의 1/2로 설정하여 계산하였다.

따라서, 수작업과 자동화 경우에 경제성 평가에 널리 활용되는 총 투자에 대한 세 가지 경제성 지표, 즉 Payback Period(시간 인자), NPV (자본 인자), 수익률(이자 인자)를 구하여 표 4.12에 정리하였다. NPV 계산에 필요한 MARR은 은행 이자율을 고려하여 10%로 설정하였다.

이런 지표들을 수작업과 비교해보면, 자동화가 세 가지 지표에 대해 모두 우수함을 파악할 수 있다.

표 4.11 연간수익계산 (2차년도 기준)

연(年)	계산 단위	수작업 (천원)	자동화 (천원)
연간 생산량	연 300일	2,430,000	5,130,000
1. 매 상 고	1개당 100원	243,000	513,000
2. 변 동 비	[기준단가]		
(1) 배양액	2원/개	4,860	10,260
(2) 세척비	0.2원/개	486	1,026
(3) 멸균비용	0.2원/개	486	1,026
(4) 전기료	1원/개	2,430	5,130
(5) 수도료	0.1원/개	243	513
소 계		8,505	17,955
3. 고 정 비	[기준단가]		
(1) 직접 노무비	1인 60만원, 100만원	43,200	12,000
(2) 고정 자산세	1% x 고정자산투자비	800	2,500
(3) 보험료	0.5% x 고정자산투자비	400	1,250
(4) 수선비	3% x 고정자산투자비, 매년 10% 상승	2,640	8,250
(5) 공장관리비	직접 노무비 x 100%	43,200	12,000
(6) 감가상각비	(정액법, 법정내용연수4년, 잔존가 0)	20,000	62,500
소 계		110,240	98,500
4. 판매관리비			
(1) 판매비	매상고 5%	5,400	12,150
(2) 일반관리비	직접노무비 100%	43,200	12,000
소 계		48,600	24,150
5. 이 자			
(1) 건설 자금	차입금 5년 균등상환, 10%/연	5,000	15,000
(2) 운전 자금	매상고 3개월분x12%/연	7,290	15,390
소 계		12,290	30,390
6. 총 원 가	(2+3+4+5)	179,635	170,995
7. 납세전 이익	매상고 - 총원가	63,365	342,005
8. 납세후 이익	법인세 17%	10,772.05	58,140.85

표 4.12 경제성 평가 (단위:천원)

경제성 지표	수작업		자동화	
	총 투자에 의한 평가	자기자본에 의한 평가	총 투자에 의한 평가	자기자본에 의한 평가
1. 시간 인자 · Payback 기간 (PBP)	3년	2년	2년	2년
2. 자본 인자 · 할인율(MARR) 10 % 의 NPV	339,464.86	341,520.52	1,682,518.60	1,688,124.95
3. 이자 인자 · 수익률 (IRR)	53.55%	70.22%	118.4%	164.55%

그리고 전술한 바와 같이 일반적으로 "총 투자에 의한 평가"로 대안을 평가하지만 사기업의 경우는 "자기자본에 의한 평가"를 보조적으로 활용하는 경우가 있으므로 여기서는 "총 투자에 의한 평가"외에 "자기자본에 의한 평가"를 세 가지 경제성 지표에 대해 계산하여 부기하였다.

- 총 투자에 의한 평가 : 부채도 초기투자비로 간주하여 이자 발생분(자본비용)을 net income에 가산하여 평가함
- 자기자본에 의한 평가 : 감가상각비와 이자발생분만 세금을 감면하는데 이용하여 자기자본 투자본에 대한 경제성을 평가함
- 경제성 평가를 위해 "**EzCash**" 프로그램(Park, C. S.가 개발한 프로그램)을 이용하였음

제 5 장 결 론

소식물체의 조직배양 방법에 의한 생산은 화훼류, 채소, 과수 분야 등에서 많이 이루어지고 있으며, 이러한 분야에 대해 앞으로 지속적인 연구개발이 행해져 새로운 형태의 첨단농업으로 발전하게 될 것이다. 첨단농업 생산은 기계화 및 자동화에 의한 높은 생산성을 달성하게 될 것이고, 또한 고부가가치 산업으로 성장하게 될 것이다. 지금까지 단순 수작업에 의해 증식작업이 행해지고 있는 조직배양실 작업 공정 및 환경에 대해 연구를 하여 첨단 자동화 기술을 적용한 조직배양실 소식물체 핸들링 시스템을 개발하였다. 본 연구에서 개발된 조직배양실 소식물체 핸들링 시스템은 첨단 농업의 발전 및 생산성 향상에 기여를 하게 될 것이다. 본 연구개발을 통해 다음과 같은 결론을 얻게 되었다.

첫째, 조직배양실 소식물체 증식작업 공정을 자동화하기 위해서는 배양방법, 배양용기, 광조명, 작업순서 등에 대한 작업환경의 표준화가 선행되어야 한다.

둘째, 본 연구개발 시스템을 통해 씨감자 증식공정에서 파지, 절단 및 이식 작업에 대해 자동화를 실현할 수 있는 것으로 나타났다. 이것은 실제 작업에 대한 적용실험을 통해 알 수 있었다.

셋째, 본 연구개발 시스템을 상품화하기 위해 관련 주변장치와 통합하여 무인화 및 자동화를 실현할 수 있도록 조직배양실의 운영방법에 대한 추가 연구 검토가 필요하다. 특히, 청정작업을 위한 관련 기술 및 시스템에 대한 연구개발이 되어야 한다. 핸들링 작업시 장치에 의한 오염이 되지 않도록 하여야 하며, 장치의 세정 방법에 대한 연구도 함께 필요하다.

넷째, 본 연구개발에 대한 경제성 평가에 의하면 가동률을 높일수록 경제성이 올라가는 것으로 나타났다. 그러므로, 많은 작업량이 나올 수 있는 대규모의 대량 생산 배양시설을 갖춘 곳에 본 연구개발 시스템을 적용할 경우 경제적으로 타당성을 갖는다.

본 연구개발에 있어서 실험용 씨감자를 제공하고 조직 배양실에 대한 여러 가지 많은 조언을 하여 주신 생명공학연구소 씨감자 연구실의 정혁 박사께 감사를 드립니다.

참고 문헌

[1] Robert M. Haralick and Linda G. Shapiro, Computer and Robot Vision, Sddison-Wesley Publishing Company, 1996.

[2] Atsushi Kinase, Yuichi Ikeda, Taizo Tateishi, Nobuyuki Fujita, "Development of the Robot System for Plantlet Factory," 제8회 일본 로봇 학회학술강연회(평성2년11월 1~3일) pp.1033-1036.

[3] Hiroaki Watake and Atsushi Kinase, "Robot for Plant Tissue Culture," Robot No. 64, pp. 74-79, 1988.

[4] 문정기, 박경택, "소식물체 생산 자동화 기술개발(I)," 과학기술처 연구보고서, 한국기계연구원, 1996.

[5] Indranil Chakravarty, "A Single-Pass, Chain Generating Algorithm for Region Boundaries," Computer Vision, Graphics, and Image Processing 15, pp.182-193, 1981.

[6] G. P. Ashkar and J. W. Modestino, "The Contour Extraction Problem with Biomedical Applications," Computer Graphics and Image Processing 7, pp. 331-355, 1978.

[7] Atsushi Kinase, Taizo Tateishi, "Development of Plantlet Micropropagation Robot," 제7회 일본 로봇 학술강연회, pp. 275-276 (평성 원년 2~4일).

[8] T. Yamamoto, M. Kitamura, A. Ito, Y. Miwa, "Automation of the

Plant Tissue Culture Process," 제4회 일본로봇 학술강연회, pp. 465-466, (소화61년12월15~17일).

[9] D. P. Holdgate and E. A. Zandvoort, "Automated Micropropagation and the Application of a Laser Beam for Cutting," Transplant Production Systems, pp. 297-311, 1992.

[10] 황헌, 이충호, "컴퓨터 시각에 의거한 측정기술 및 측정오차의 분석과 보정," 한국농업기계학회지, 제17권 제1호, 1992년 3월.

[11] 이종환, 노상하, "농산물 및 미립자의 기하학적 특성 분석을 위한 컴퓨터 시각 시스템(I,II)," 한국농업기계학회, 제17권 제2호, 1992년 6월.

[12] Seong-Dae Kim, Jeong-Hwan Lee, and Jae-Kyoon Kim, " A New Chain-Coding Algorithm for Binary Images Using Run-Length Codes," Computer Vision, Graphics and Image Processing 42, pp. 114-128, 1988.

[13] Jean Paul Lauzon, David M. Mark, Lawrence Kikuchi, "Two-Dimensional Run-Encoding for Quadtree Representation," Computer Vision, Graphics, and Image Processing 30, pp. 56-69, 1985.

[14] 박주철, "첨단생산시스템을 위한 기술원가 모델의 개발," 산업공학, 제8권, 제1호, 1995.

[15] 김성인, 김승권, "유연 생산 시스템 도입의 경제적 타당성 평가를 위한 전략적 모델," 산업공학, 제1권, 제2호, 1988.

[16] 박경수, 경제성 공학, 구민사, 1997.

[17] 유일근, 원가측정과 분석, 시그마프레스, 1997.

[18] Park, C. S., Contemporary Engineering Economics, 2nd edition, Addison-Wesley, 1997.

[19] 유일근, 최신경제성공학, 형설출판사, 1998.

여 백

부 록

1. 시스템 관련 설계도면 및 각종 자료

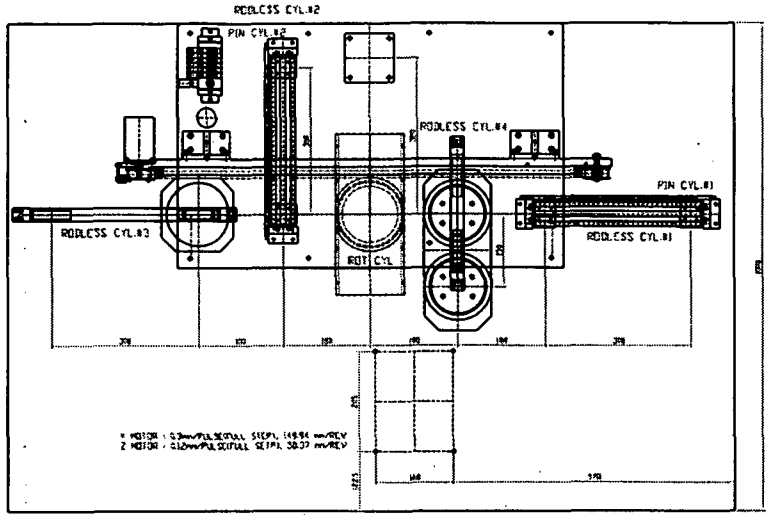
2. 시스템 관련 S/W Source

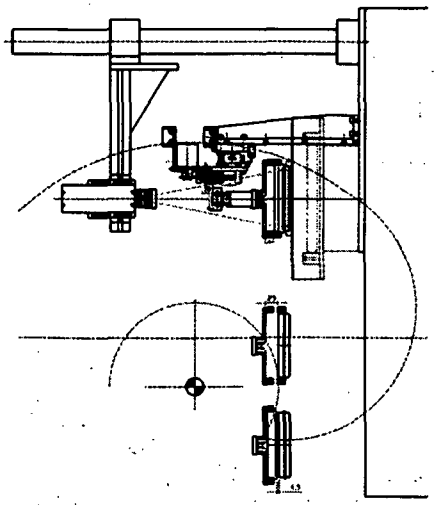
여 백

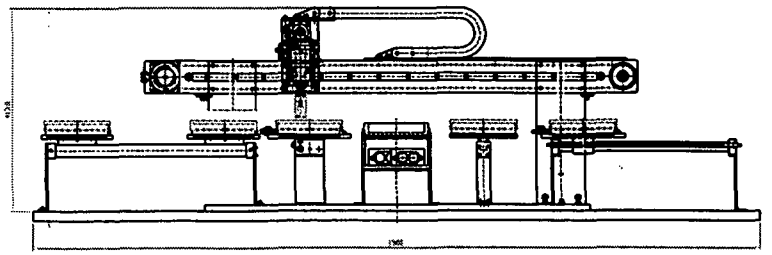
1. 시스템 관련 설계도면 및 각종 자료

여 백

Rev No	Revision Note	Date	Signature	Checked
△				
△				





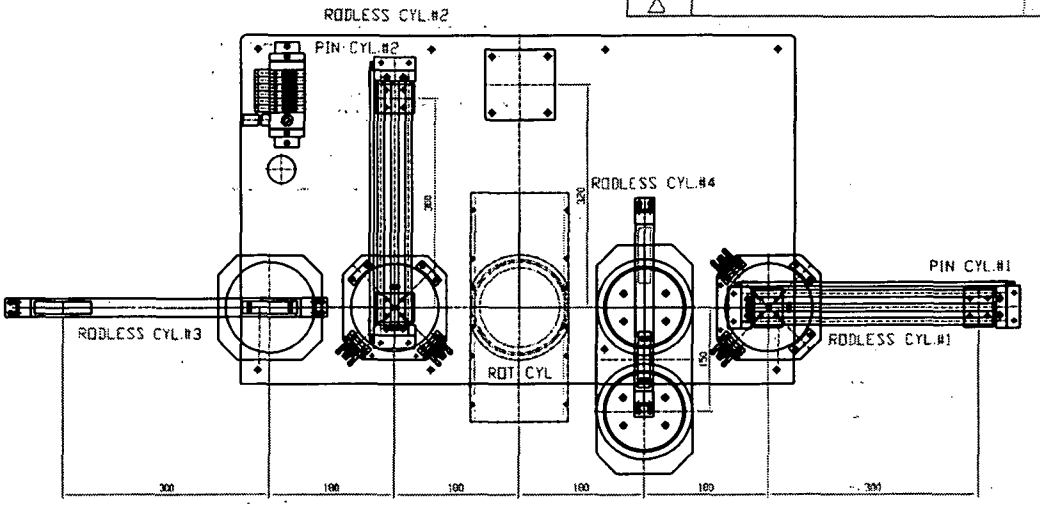


No	Drawing Name	Material	Q'ty	Finish
000	POTATO M/C ASS'Y	–	1	–

KIMM	Drawn	Check	Approv.	Title	POTATO M/C ASS'Y
	MWPARK 990901				POTATO M/C.

Unit	Scale	Finish Layout	Dwg.no	Sheet
mm	1/1		POT99 – 000	A3

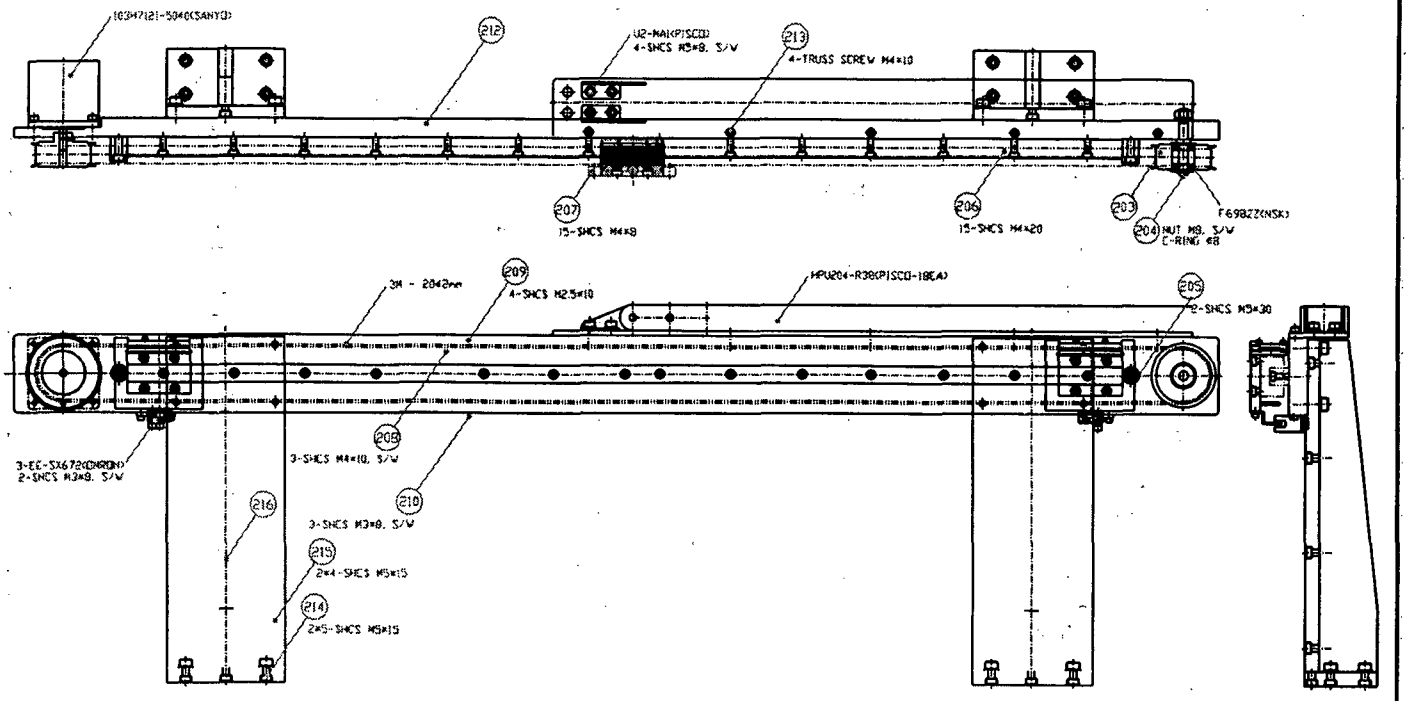
Rev No	Revision Note	Date	Signature	Checked
△				
△				



No	Drawing Name	Material	Q'ty	Finish
300	FEEDER PART ASSY	-	1	-

KIMM	Drawn	Check	Approv.	Title	FEEDER PART POTATO M/C
	MWPARK 990901				
Unit: mm	Scale: 1/1	Finish Layout		Dwg.no: POT99 – 300	Sheet: A3

Rev No	Revision Note	Date	Signature	Checked
△				
△				

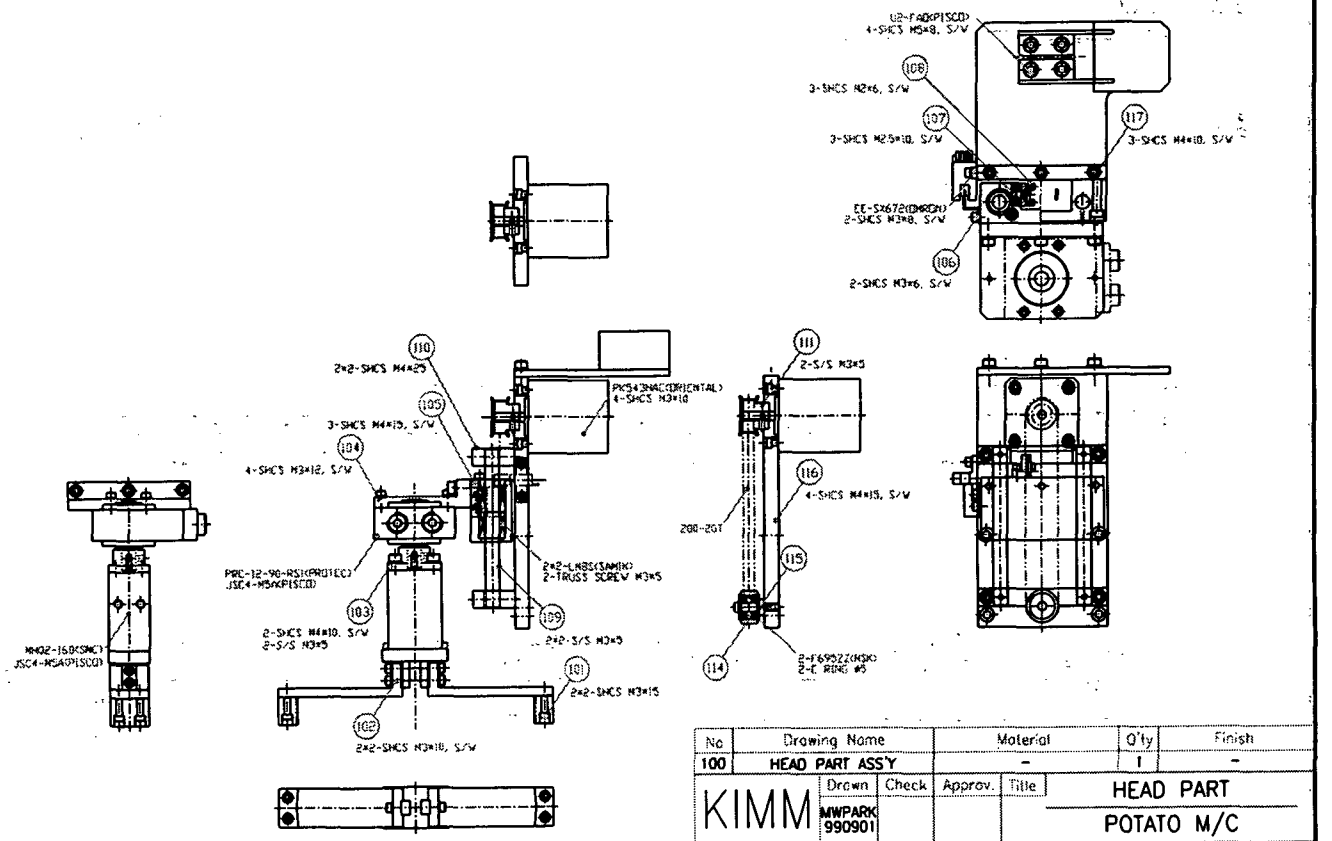


No	Drawing Name	Material	Q'ty	Finish
200	HANDLER PART ASS'Y	-	1	-

KIMM	Drawn	Check	Approv.	Title
	MWPARK 990901			HANDLER PART POTATO M/C

Unit	Scale	Finish Layout	Dwg.no	Sheet
mm	1/1	-	POT99 – 200	A3

Rev No	Revision Note	Date	Signature	Checked
△				
△				

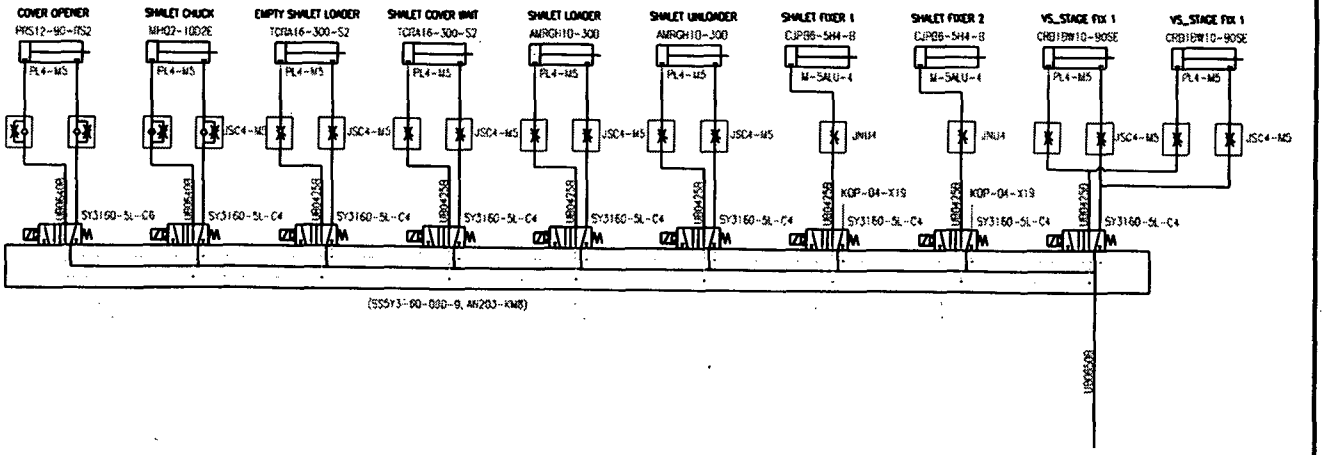


No	Drawing Name	Material	Q'ty	Finish
100	HEAD PART ASS'Y	-	1	-

KIMM	Drawn	Check	Approv.	Title	HEAD PART POTATO M/C
	MWPARK 990901				

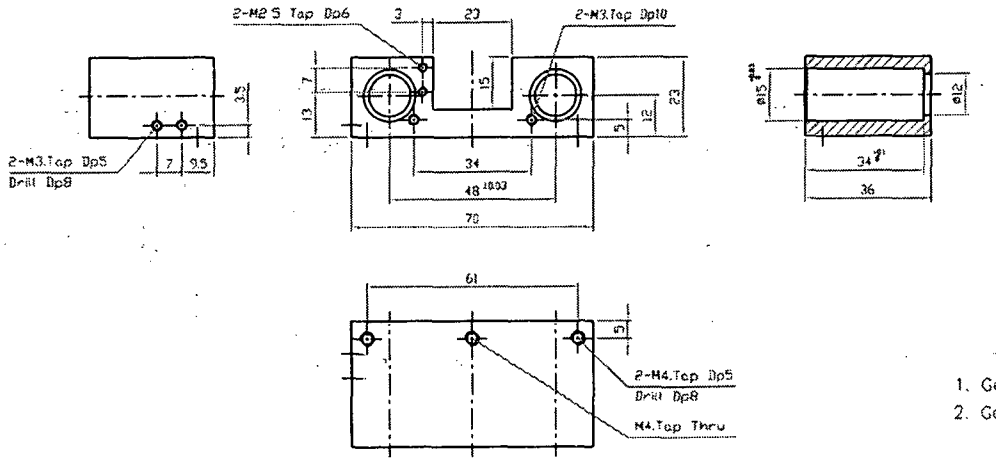
Unit	Scale	Finish Layout	Dwg.no	Sheet
mm	1/1	-	POT99 - 100	A3

Rev No	Revision Note	Date	Signature	Checked
△				
△				



No	Drawing Name		Material		Q'ty	Finish
400	AIR - FLOW		-		1	-
KIMM	Drawn	Check	Approv.	Title	AIR FLOW	
	MWPARK 990901				POTATO M/C	
Unit	Scale	Finish Layout		Dwg no	POT99 - 400	Sheet
mm	1/1	-				A3

Rev No	Revision Note	Date	Signature	Checked
△				
△				



NOTES

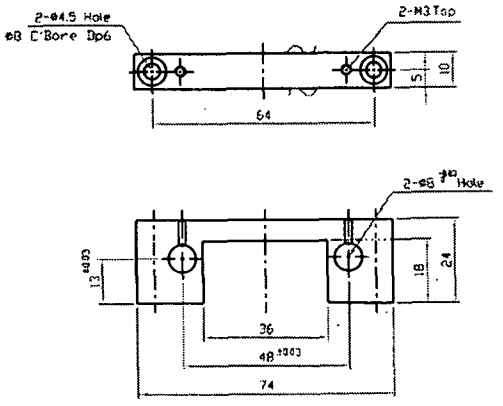
1. General Chamfer : C0.2 ~0.5
2. General Tolerance : ±0.1

No	Drawing Name	Material	Q'ty	Finish
105	CARRIER	A6061-T6	1	WHITE ANODIZING

KIMM	Drawn	Check	Approv.	Title	HEAD PART POTATO M/C
	MWPARK 990901				

Unit	Scale	Finish Layout	Dwg.no	Sheet
mm	1/1	70 • 36 • 23	POT99 – 105	A3

Rev No	Revision Note	Date	Signature	Checked
△				
△				



NOTES

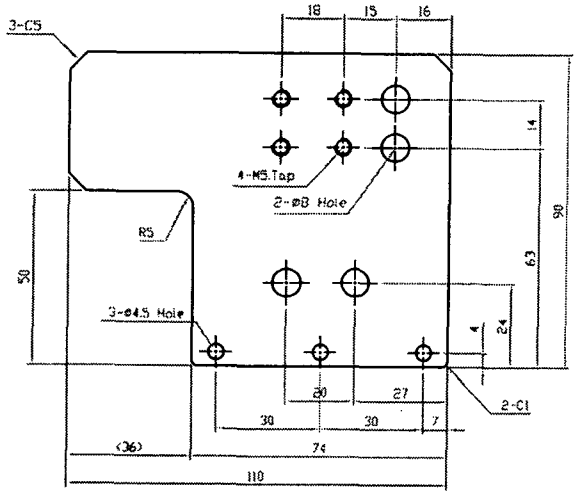
1. General Chamfer : C0.2 ~0.5
2. General Tolerance : ±0.1

No	Drawing Name	Material	Q'ty	Finish
110	BRACKET - SHAFT	A5052P 10t	2	WHITE ANODIZING

KIMM	Drawn	Check	Approv	Title	HEAD PART
	MWPARK 990901				POTATO M/C

Unit	Scale	Finish Layout	Dwg no	Sheet
mm	1/1	74 * 24 * 10	POT99 - 110	A3

Rev No	Revision Note	Date	Signature	Checked
△				
△				



t=4mm

NOTES

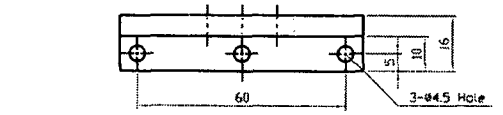
1. General Chamfer : C0.2 ~0.5
2. General Tolerance : ±0.1

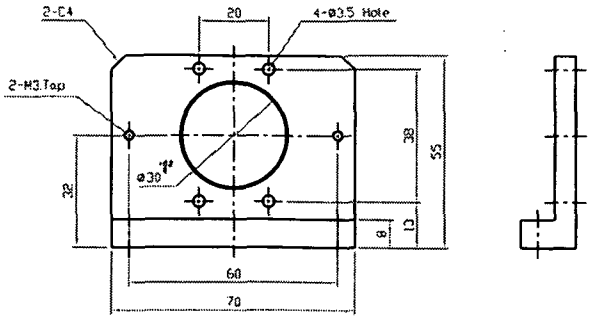
No	Drawing Name	Material	Q'ty	Finish
117	BRACKET - CHAIN	A5052P 4t	1	WHITE ANODIZING

KIMM	Drawn	Check	Approv.	Title	HEAD PART
	MWPARK 990901				POTATO M/C

Unit	Scale	Finish Layout	Dwg.no	Sheet
mm	1/1	110 * 90 * 4	POT99 - 117	A3

Rev No	Revision Note	Date	Signature	Checked
△				
△				





NOTES

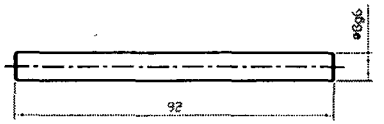
1. General Chamfer : C0.2 ~0.5
2. General Tolerance : ±0.1

No	Drawing Name	Material	Q'ty	Finish
104	BRACKET - HAND	A6061-T6	1	WHITE ANODIZING

KIMM	Drawn	Check	Approv.	Title
	MWPARK 990901			HEAD PART POTATO M/C

Unit	Scale	Finish Layout	Dwg.no	Sheet
mm	1/1	70 * 55 * .16	POT99 - 104	A3

Rev No	Revision Note	Date	Signature	Checked
1				
2				



NOTES

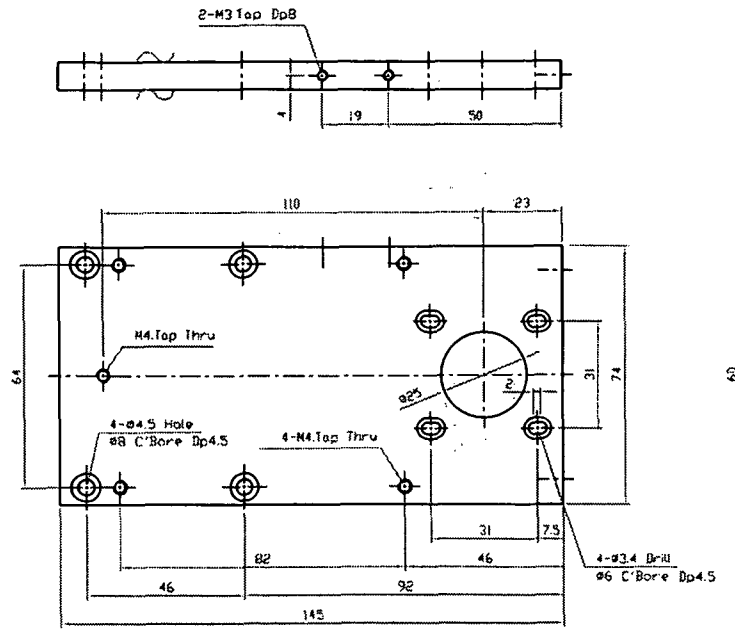
1. General Chamfer : C0.2 ~0.5
2. General Tolerance : ±0.1

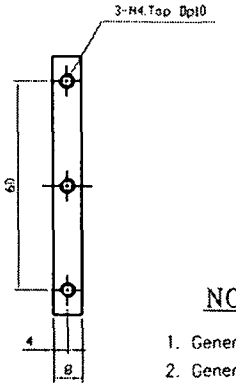
No	Drawing Name	Material	Q'ty	Finish
109	SHAFT-HEAD	SUS304 ø8	2	-

KIMM	Drawn	Check	Approv.	Title
	MWPARK 990901			HEAD PART POTATO M/C

Unit	Scale	Finish Layout	Dwg.no	Sheet
mm	1/1	ø8 * 92	POT99 - 109	A3

Rev No	Revision Note	Date	Signature	Checked
△				
△				





NOTES

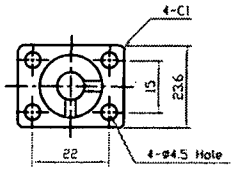
1. General Chamfer : C0.2 ~0.5
2. General Tolerance : ±0.1

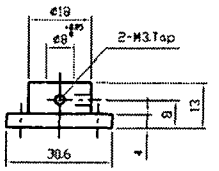
No	Drawing Name	Material	Q'ty	Finish
116	BASE HEAD	A5052P 8t	1	WHITE ANODIZING

KIMM	Drawn	Check	Approv.	Title	HEAD PART POTATO M/C
	MWPARK 990901				

Unit	Scale	Finish Layout	Dwg.no	Sheet
mm	1/1	145 • 74 • 8	POT99 - 116	A3

Rev No	Revision Note	Date	Signature	Checked
△				
△				





NOTES

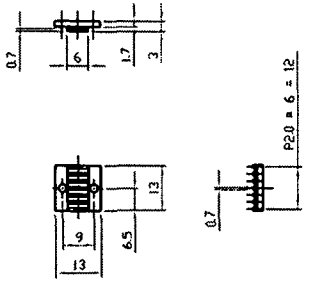
1. General Chamfer : C0.2 ~0.5
2. General Tolerance : ±0.1

No	Drawing Name	Material	Q'ty	Finish
103	JOINT	A6061-T6	1	WHITE ANODIZING

KIMM	Drawn	Check	Approv.	Title
	MWPARK 990901			HEAD PART POTATO M/C

Unit	Scale	Finish Layout	Dwg.no	Sheet
mm	1/1	30.6 • 23.6 • 13	POT99 - 103	A3

Rev No	Revision Note	Date	Signature	Checked
△				
△				



NOTES

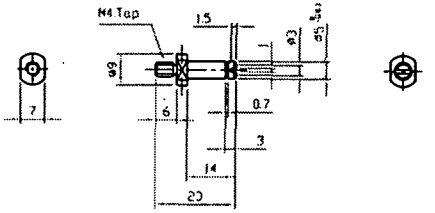
1. General Chamfer : C0.2 ~0.5
2. General Tolerance : ±0.1

No	Drawing Name	Material	Q'ty	Finish
108	FIXER2 - BELT	A6061-T6	1	WHITE ANODIZING

KIMM	Drawn	Check	Approv.	Title
	MWPARK 990901			HEAD PART POTATO M/C

Unit	Scale	Finish Layout	Dwg.no	Sheet
mm	1/1	13 * 13 ±.3	POT99 - 108	A3

Rev No	Revision Note	Date	Signature	Checked
△				
△				



NOTES

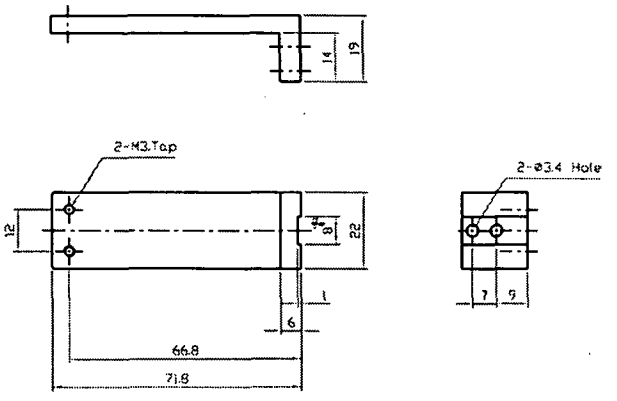
1. General Chamfer : C0.2 ~0.5
2. General Tolerance : ±0.1

No	Drawing Name	Material	Q'ty	Finish
115	SHAFT - PULLEY	SM45C	1	Ni PLATE

KIMM	Drawn	Check	Approv.	Title	HEAD PART POTATO M/C
	MWPARK 990901				

Unit	Scale	Finish Layout	Dwg.no	Sheet
mm	1/1	ø9 * 23	POT99 - 115	A3

Rev No	Revision Note	Date	Signature	Checked
△				
△				



NOTES

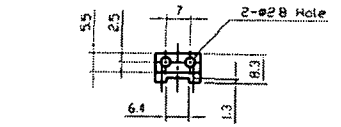
1. General Chamfer : C0.2 ~0.5
2. General Tolerance : ±0.1

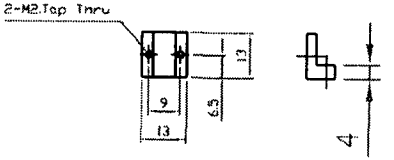
No	Drawing Name	Material	Q'ty	Finish
102	BRACKET - HOLDER	A6061-T6	2	WHITE ANODIZING

KIMM	Drawn	Check	Approv.	Title
	MWPARK 990901			HEAD PART POTATO M/C

Unit	Scale	Finish Layout	Dwg.no	Sheet
mm	1/1	71.8 * 22 * 19	POT99 - 102	A3

Rev No	Revision Note	Date	Signature	Checked
△				
△				





NOTES

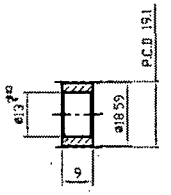
1. General Chamfer : C0.2 ~0.5
2. General Tolerance : ±0.1

No	Drawing Name	Material	Q'ty	Finish
107	FIXER1 – BELT	A6061-T6	1	WHITE ANODIZING

KIMM	Drawn	Check	Approv.	Title
	MWPARK 990901			HEAD PART POTATO M/C

Unit	Scale	Finish Layout	Dwg.no	Sheet
mm	1/1	13 • 13 • 8.3	POT99 – 107	A3

Rev No	Revision Note	Date	Signature	Checked
△				
△				



NOTES

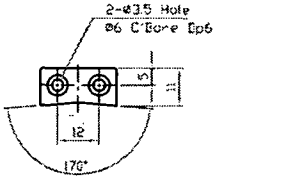
1. General Chamfer : C0.2 ~0.5
2. General Tolerance : ±0.1

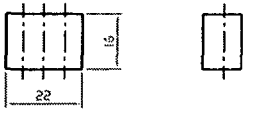
No	Drawing Name	Material	Q'ty	Finish
114	PULLEY - SHAFT	A6061(30-2GT)	1	WHITE ANODIZING

KIMM	Drawn	Check	Approv.	Title
	MWPARK 990901			HEAD PART POTATO M/C

Unit	Scale	Finish Layout	Dwg.no	Sheet
mm	1/1	ø18.59 * 9	POT99 - 114	A3

Rev No	Revision Note	Date	Signature	Checked
△				
△				





NOTES

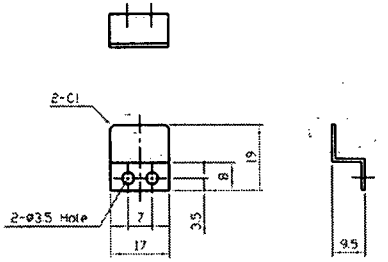
1. General Chamfer : C0.2 ~0.5
2. General Tolerance : ±0.1

No	Drawing Name	Material	Q'ty	Finish
101	HOLDER	URETHANE	2	-

KIMM	Drawn	Check	Approv.	Title	HEAD PART
	MWPARK 990901				POTATO M/C

Unit	Scale	Finish Layout	Dwg.no	Sheet
mm	1/1	22 • 16 • 11	POT99 - 101	A3

Rev No	Revision Note	Date	Signature	Checked
△				
△				



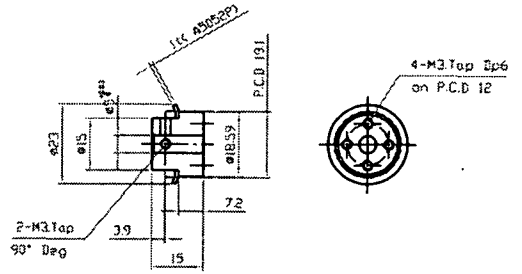
t=1.0

NOTES

1. General Chamfer : C0.2 ~0.5
2. General Tolerance : ±0.1

No	Drawing Name		Material	Q'ty	Finish
106	DOG - SENSOR		STS304	1	-
KIMM	Drawn: MWPARK 990901	Check	Approv.	Title: HEAD PART POTATO M/C	
Unit: mm	Scale: 1/1	Finish Layout: 19 * 17 * 9.5		Dwg.no: POT99 - 106	Sheet: A3

Rev No	Revision Note	Date	Signature	Checked
△				
△				



NOTES

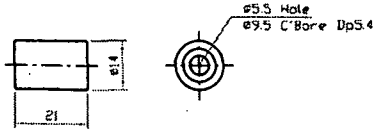
1. General Chamfer : C0.2 ~0.5
2. General Tolerance : ±0.1

No	Drawing Name	Material	Q'ty	Finish
111	PULLEY - MOTOR	A6061(30-2GT)	1	WHITE ANODIZING

KIMM	Drawn	Check	Approv.	Title
	MWPARK 990901			HEAD PART POTATO M/C

Unit	Scale	Finish Layout	Dwg.no	Sheet
mm	1/1	ø23 • 15	POT99 - 111	A3

Rev No	Revision Note	Date	Signature	Checked
△				
△				



NOTES

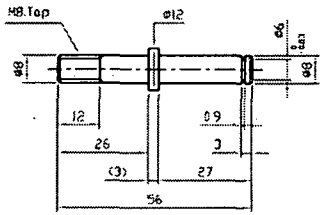
1. General Chamfer : C0.2 ~0.5
2. General Tolerance : ±0.1

No	Drawing Name	Material	Q'ty	Finish
205	STOPPER	URETHANE	1	-

KIMM	Drawn	Check	Approv.	Title	HANDLER PART
	MWPARK 990901				POTATO M/C

Unit	Scale	Finish Layout	Dwg.no	Sheet
mm	1/1	ø14 * 21	POT99 - 205	A3

Rev No	Revision Note	Date	Signature	Checked
△				
△				



NOTES

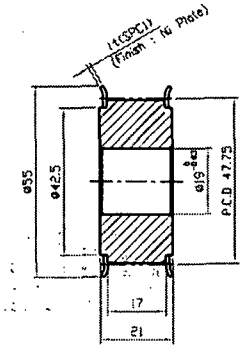
1. General Chamfer : C0.2 ~0.5
2. General Tolerance : ±0.1

No	Drawing Name	Material	Q'ty	Finish
204	SHAFT - PULLEY	SM45C	1	Ni PLATE

KIMM	Drawn	Check	Approv.	Title
	MWPARK 990901			HANDLER PART POTATO M/C

Unit	Scale	Finish Layout	Dwg.no	Sheet
mm	1/1	ø12 * 56	POT99 - 204	A3

Rev No	Revision Note	Date	Signature	Checked
△				
△				



NOTES

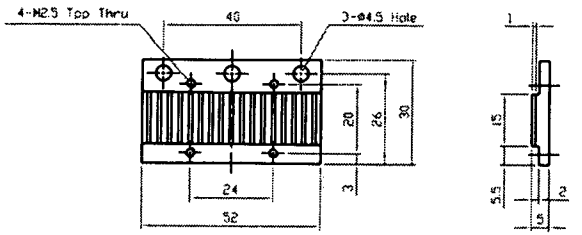
1. General Chamfer : C0.2 ~0.5
2. General Tolerance : ±0.1

No	Drawing Name	Material	Q'ty	Finish
203	PULLEY - SHAFT	A6061 (5OP-3M-15BF)	1	WHITE ANODIZING

KIMM	Drawn	Check	Approv.	Title
	MWPARK 990901			HANDLER PART POTATO M/C

Unit	Scale	Finish Layout	Dwg.no	Sheet
mm	1/1	ø55 * 21	POT99 - 203	A3

Rev No	Revision Note	Date	Signature	Checked
△				
△				



NOTES

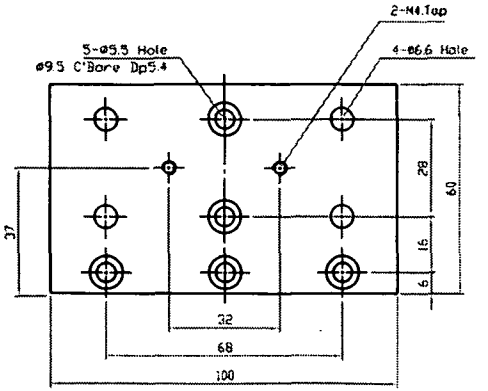
1. General Chamfer : C0.2 ~0.5
2. General Tolerance : ±0.1

No	Drawing Name	Material	Q'ty	Finish
208	FIX1 - BELT	A6061-T6	1	WHITE ANODIZING

KIMM	Drawn	Check	Approv.	Title	HANDLER PART
	MWPARK 990901				POTATO M/C

Unit	Scale	Finish Layout	Dwg.no	Sheet
mm	1/1	52 • 30 • 5	POT99 - 208	A3

Rev No	Revision Note	Date	Signature	Checked
△				
△				



NOTES

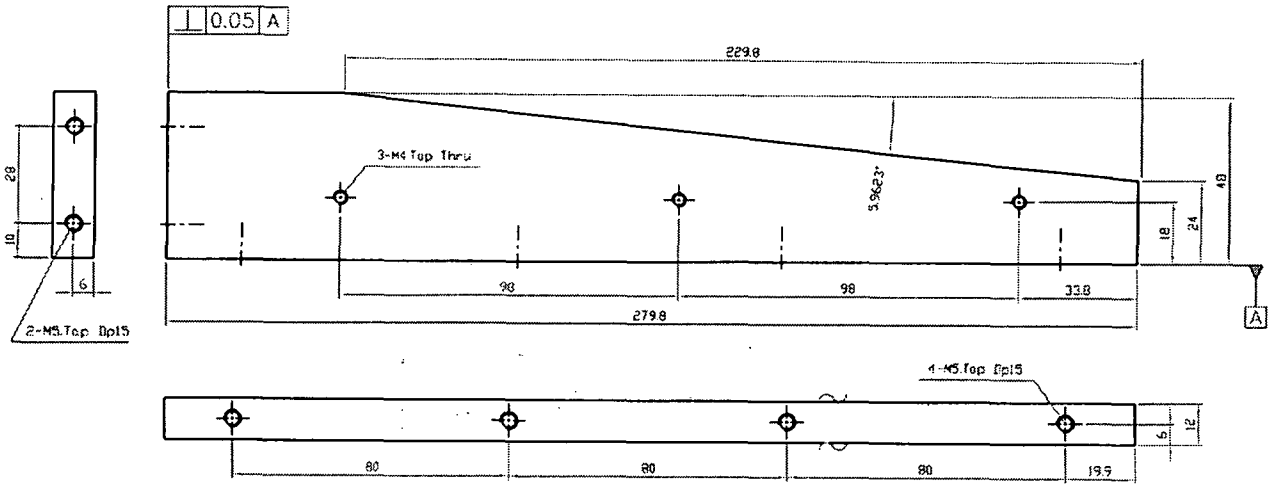
1. General Chamfer : C0.2 ~0.5
2. General Tolerance : ±0.1

No	Drawing Name	Material	Q'ty	Finish
216	BASE 3	A5052P 12t	2	WHITE ANODIZING

KIMM	Drawn	Check	Approv.	Title
	MWPARK 990901			HANDLER PART POTATO M/C

Unit	Scale	Finish Layout	Dwg.no	Sheet
mm	1/1	100 * 60 * 12t	POT99 - 216	A3

Rev No	Revision Note	Date	Signature	Checked
△				
△				



NOTES

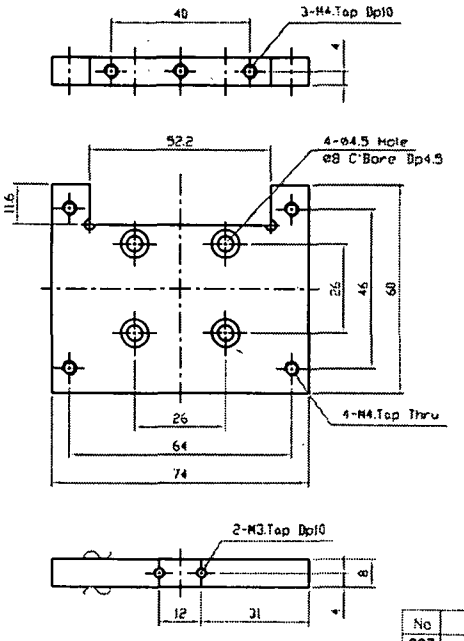
1. General Chamfer : C0.2 ~0.5
2. General Tolerance : ±0.1

No	Drawing Name	Material	Q'ty	Finish
215	BASE 2	A5052P 12t	2	WHITE ANODIZING.

KIMM	Drawn	Check	Approv.	Title
	MWPARK 990901			HANDLER PART POTATO M/C

Unit	Scale	Finish Layout	Dwg.no	Sheet
mm	1/1	279.8 • 48 • 12t.	POT99 – 215	A3

Rev No	Revision Note	Date	Signature	Checked
△				
△				



NOTES

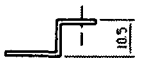
1. General Chamfer : C0.2 ~0.5
2. General Tolerance : ±0.1

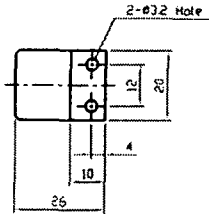
No	Drawing Name	Material	Q'ty	Finish
207	CARRIER	A5052P 8t	1	WHITE ANODIZING

KIMM	Drawn	Check	Approv.	Title
	MWPARK 990901			HANDLER PART POTATO M/C

Unit	Scale	Finish Layout	Dwg.no	Sheet
mm	1/1	74 * 60 * 8	POT99 - 207	A3

Rev No	Revision Note	Date	Signature	Checked
△				
△				





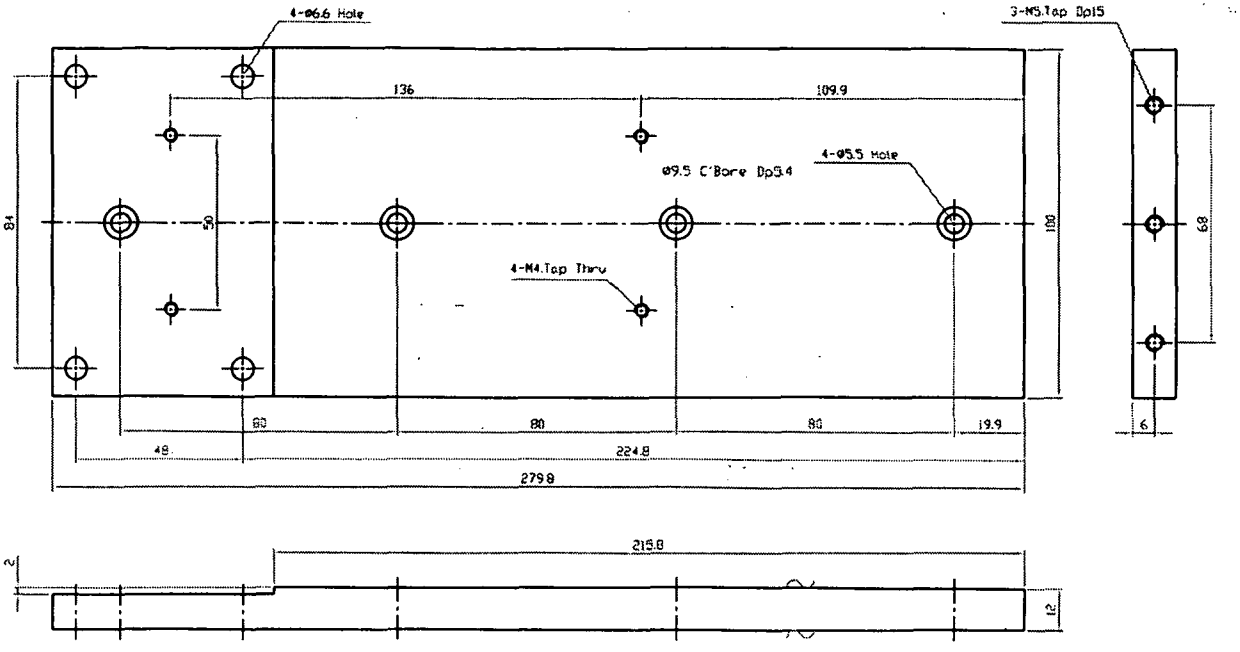
t=1mm

NOTES

1. General Chamfer : C0.2 ~0.5
2. General Tolerance : ±0.1

No	Drawing Name		Material		Q'ty	Finish
210	DOG - SENSOR		STS304 1t		1	-
KIMM	Drawn	Check	Approv.	Title	HANDLER PART	
	MWPARK 990901				POTATO M/C	
Unit	Scale	Finish Layout		Dwg.no		Sheet
mm	1/1	26 • 20 • 10.5			POT99 - 210	A3

Rev No	Revision Note	Date	Signature	Checked
△				
△				

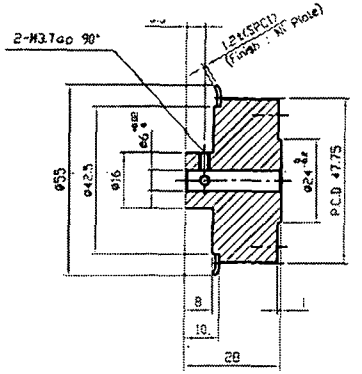


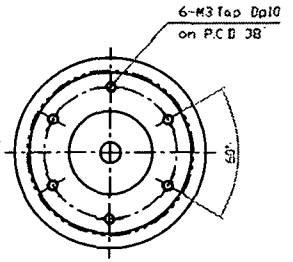
NOTES

1. General Chamfer : C0.2 ~0.5
2. General Tolerance : ±0.1

No	Drawing Name		Material	Q'ty	Finish
214	BASE 1		A5052P 12t	2	WHITE ANODIZING
KIMM	Drawn	Check	Approv.	Title	HANDLER PART
	MWPARK 990901				POTATO M/C
Unit	Scale	Finish Layout		Dwg no	Sheet
mm	1/1	279.8 * 100 * 12t		POT99 - 214	A3

Rev No	Revision Note	Date	Signature	Checked
△				
△				





NOTES

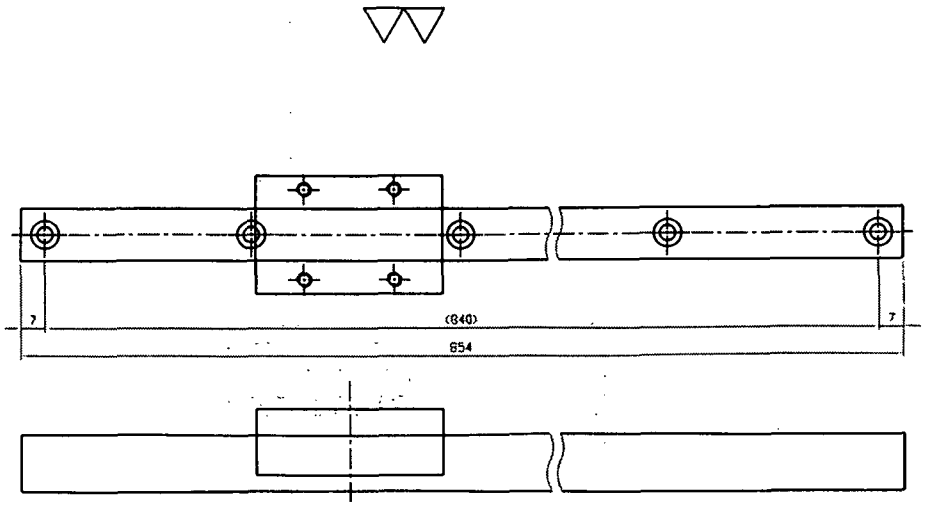
1. General Chamfer : C0.2 ~0.5
2. General Tolerance : ±0.1

No	Drawing Name	Material	Q'ty	Finish
201	PULLEY - MOTOR	A6061 (50P-3M-15BF)	1	WHITE ANODIZING

KIMM	Drawn	Check	Approv.	Title
	MWPARK 990901			HANDLER PART POTATO M/C

Unit	Scale	Finish Layout	Dwg.no	Sheet
mm	1/1	ø55 * 28	POT99 - 201	A3

Rev No	Revision Note	Date	Signature	Checked
△				
△				



NOTES

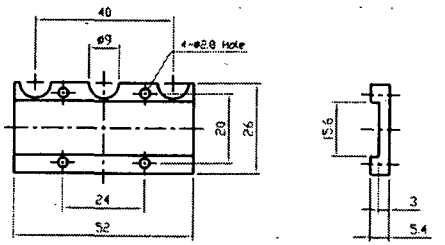
1. General Chamfer : C0.2 ~0.5
2. General Tolerance : ±0.1

No	Drawing Name	Material	Q'ty	Finish
206	LM GUIDE	SUJ2(1622-193-10, STAR)	1	-

KIMM	Drawn	Check	Approv.	Title
	MWPARK 990901			HANDLER PART POTATO M/C

Unit	Scale	Finish Layout	Dwg.no	Sheet
mm	1/1	854 * 19.4 * 15	POT99 - 206	A3

Rev No	Revision Note	Date	Signature	Checked
△				
△				



NOTES

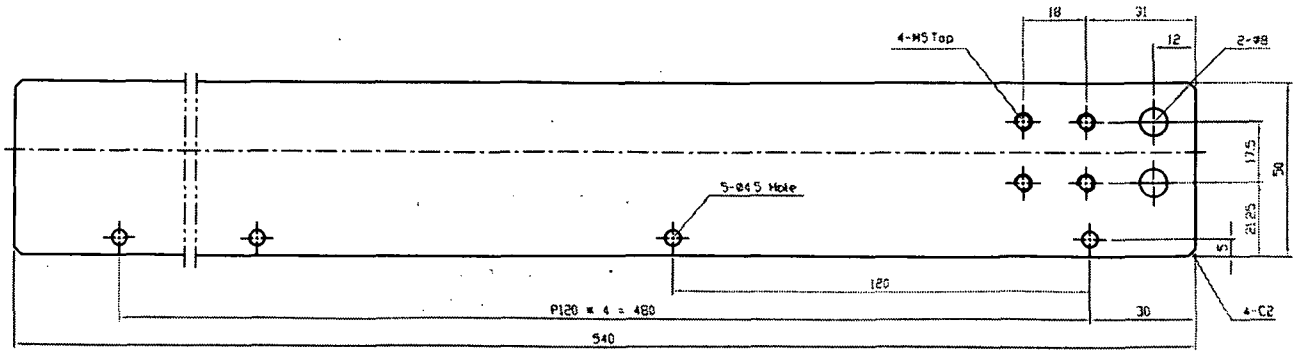
1. General Chamfer : C0.2 ~0.5
2. General Tolerance : ±0.1

No	Drawing Name	Material	Q'ty	Finish
209	FIX2 – BELT	A6061-T6	1	WHITE ANODIZING

KIMM	Drawn	Check	Approv.	Title
	MWPARK 990901			HANDLER PART POTATO M/C

Unit	Scale	Finish Layout	Dwg.no	Sheet
mm	1/1	52 * 26 * 5.4	POT99 – 209	A3

Rev No	Revision Note	Date	Signature	Checked
△				
△				



t=3mm

NOTES

1. General Chamfer : C0.2 ~0.5
2. General Tolerance : ±0.1

No	Drawing Name		Material		Q'ty	Finish
213	BRACKET - CHAIN		A5052P 3t		1	-
KIMM	Drawn	Check	Approv.	Title	HANDLER PART	
	MWPARK 990901				POTATO M/C	
Unit	Scale	Finish Layout		Dwg.no	POT99 - 213	Sheet
mm	1/1	540 * 50 * 3				A3

Rev No	Revision Note	Date	Signature	Checked
△				
△				

FREE LENGTH : 22mm

OUT DIAMETER : 6.13MM

INSIDE DIAMETER : 5.25mm

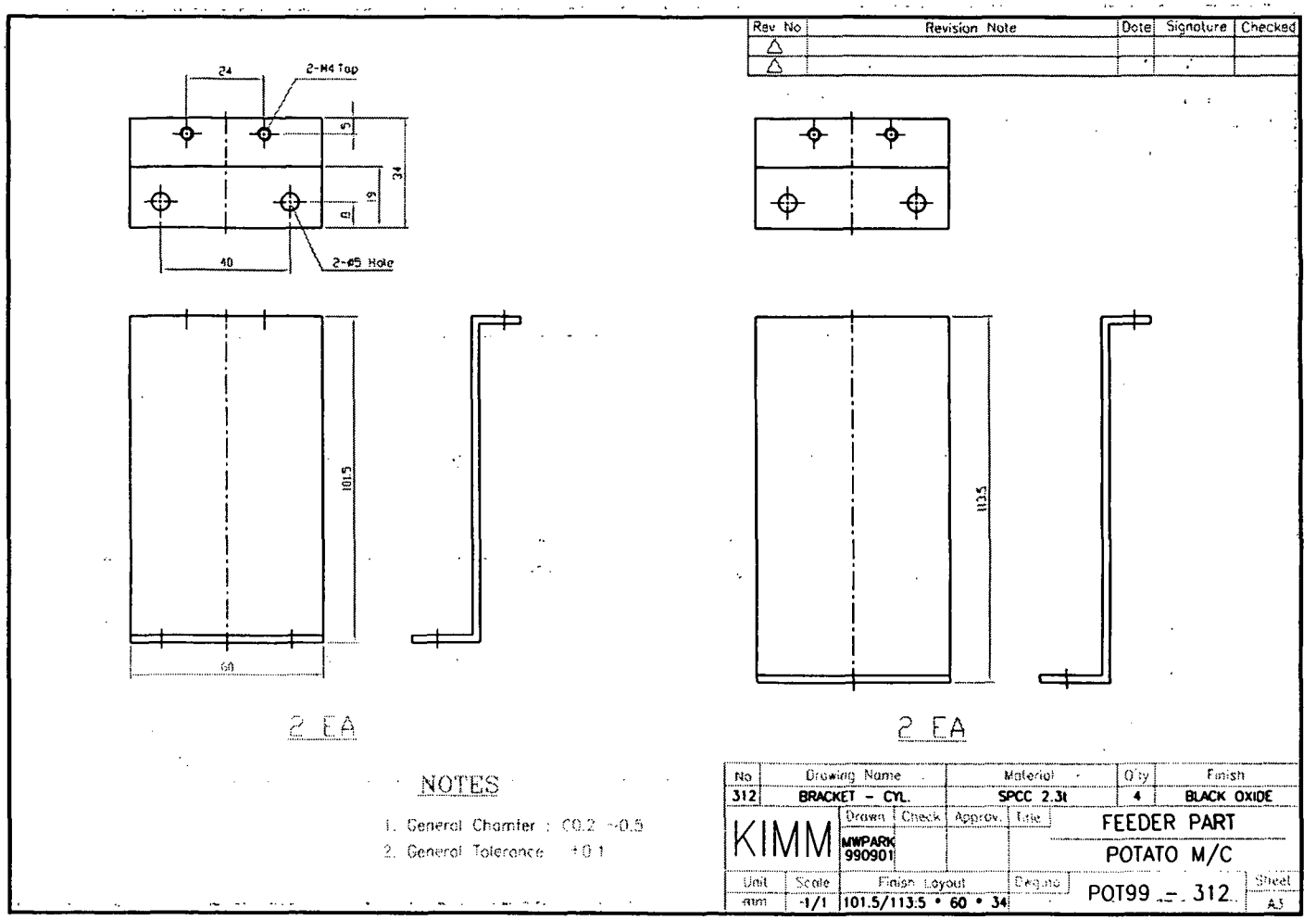
MAT'L DIAMETER : 0.44mm

ROTATION NO : 10 ROT

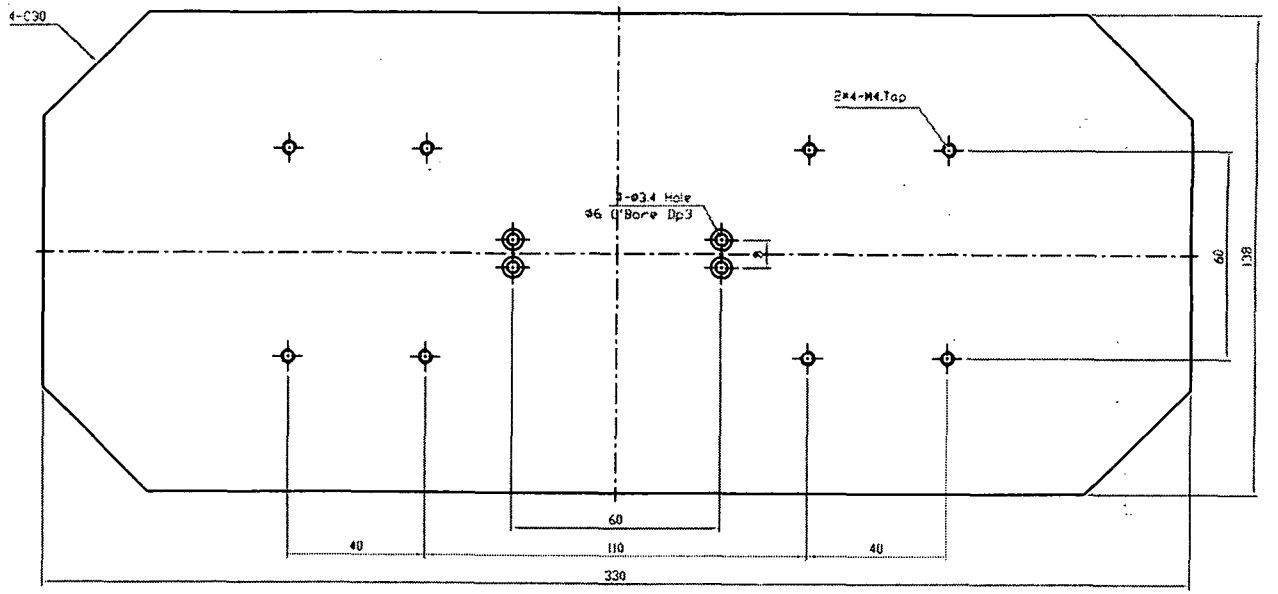
NOTES

1. General Chamfer : C0.2 ~0.5
2. General Tolerance : ±0.1

No	Drawing Name		Material	Q'ty	Finish
305	SPRING		STS304-WPB	4	-
KIMM	Drawn	Check	Approv.	Title	FEEDER PART
	MWPARK 990901				POTATO M/C
Unit	Scale	Finish Layout		Dwg.no	Sheet
mm	1/1	ϕ6.37 * 22		POT99 - 305	A3


Rev No
Revision Note
Date
Signature
Checked
24
2-M4 Tap
5
34
19
8
40
2-ø5 Hole
101.5
60
113.5
2 EA
2 EA
NOTES
1. General Chamfer : C0.2 ~0.5
2. General Tolerance : ±0.1
No
Drawing Name
Material
Q'ty
Finish
312
BRACKET - CYL.
SPCC 2.3t
4
BLACK OXIDE
KIMM
Drawn
Check
Approv.
Title
FEEDER PART
POTATO M/C
MWPARK
990901
Unit
Scale
Finish Layout
Designs
Sheet
mm
-1/1
101.5/113.5 • 60 • 34
POT99 _-_ 312
A3

Rev No	Revision Note	Date	Signature	Checked
△				
△				



NOTES

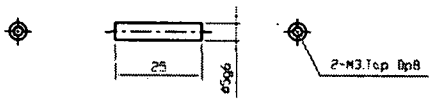
1. General Chamfer : C0.2 ~0.5
2. General Tolerance : ±0.1

No	Drawing Name	Material	Q'ty	Finish
316	PLATE	A5052P 8t	1	WHITE ANODIZING

KIMM	Drawn	Check	Approv.	Title
	MWPARK 990901			FEEDER PART POTATO M/C

Unit	Scale	Finish Layout	Dwg.no	Sheet
mm	1/1	290 • 138 • 8	POT99 – 316	A3

Rev No	Revision Note	Date	Signature	Checked
△				
△				



NOTES

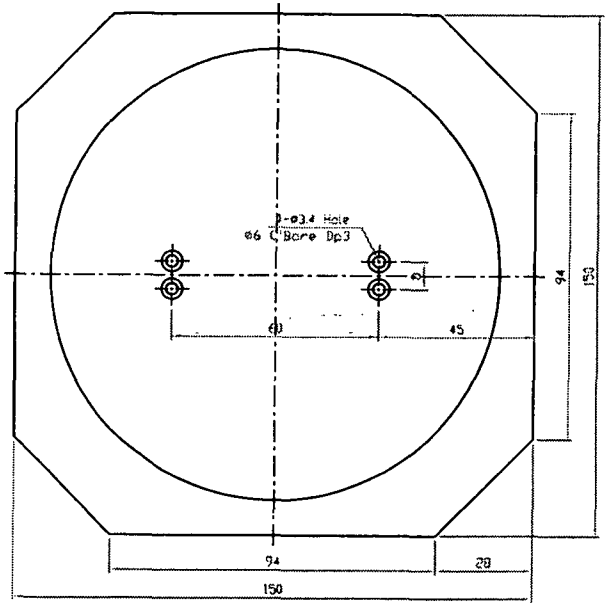
1. General Chamfer : C0.2 ~0.5
2. General Tolerance : ±0.1

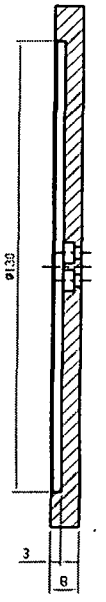
No	Drawing Name	Material	Q'ty	Finish
304	SHAFT	SUS304 ø5	4	-

KIMM	Drawn	Check	Approv.	Title	FEEDER PART POTATO M/C
	MWPARK 990901				

Unit	Scale	Finish Layout	Dwg.no	Sheet
mm	1/1	ø5 * 25	POT99 - 304	A3

Rev No	Revision Note	Date	Signature	Checked
△				
△				





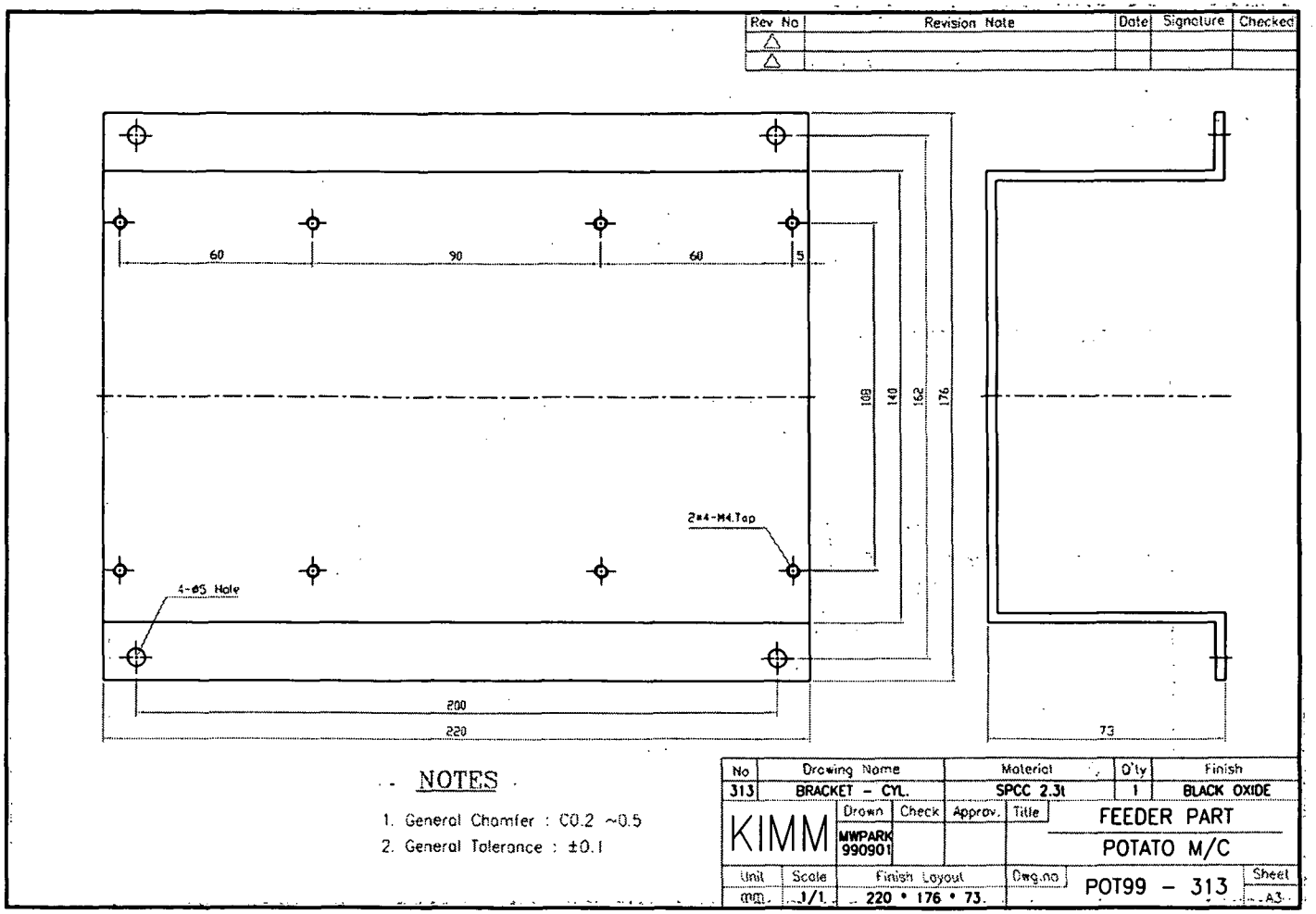
NOTES

1. General Chamfer : C0.2 ~0.5
2. General Tolerance : ±0.1

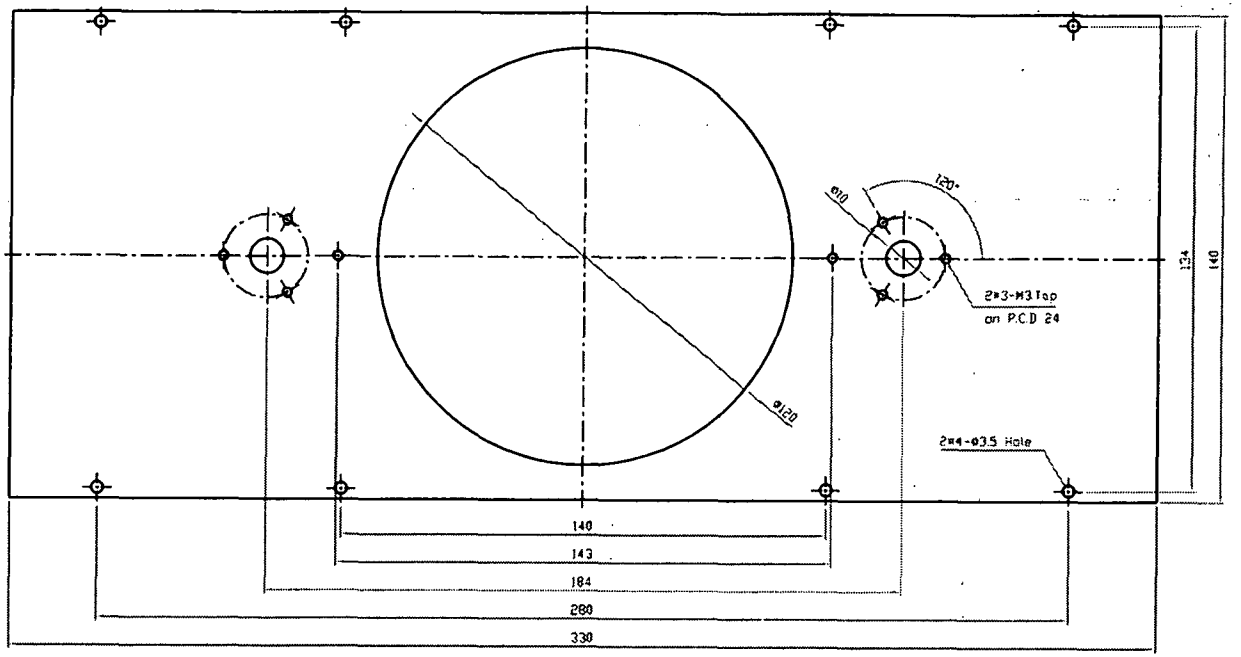
No	Drawing Name	Material	Q'ty	Finish
311	BASE - SHALET	MC NYLON (WHITE)	1	-

KIMM	Drawn	Check	Approv.	Title
	MWPARK 990901			FEEDER PART POTATO M/C

Unit	Scale	Finish Layout	Dwg.no	Sheet
mm	1/1	150 • 150 • 8t	POT99 - 311	A3


Rev No
Revision Note
Date
Signature
Checked
60
90
60
5
108
140
162
176
2-M4 Tap
4-Ø5 Hole
200
220
73
NOTES
1. General Chamfer : C0.2 ~0.5
2. General Tolerance : ±0.1
No
313
Drawing Name
BRACKET - CYL.
Material
SPCC 2.3t
Q'ty
1
Finish
BLACK OXIDE
KIMM
Drawn
MWPARK
990901
Check
Approv.
Title
FEEDER PART
POTATO M/C
Unit
mm
Scale
1/1
Finish Layout
220 • 176 • 73
Dwg.no
POT99 - 313
Sheet
A3

Rev No	Revision Note	Date	Signature	Checked
△				
△				



NOTES

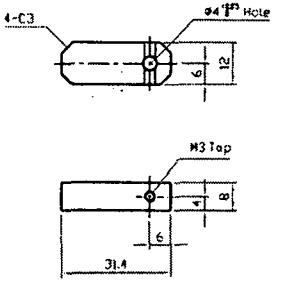
1. General Chamfer : C0.2 ~0.5
2. General Tolerance : ±0.1

No	Drawing Name	Material	Q'ty	Finish
314	PLATE	A5052P 3t	1	WHITE ANODIZING

KIMM	Drawn	Check	Approv.	Title
	MWPARK 990901			FEEDER PART POTATO M/C

Unit	Scale	Finish Layout	Dwg.no	Sheet
mm	1/1	330 * 130 * 3	POT99 - 314	A3

Rev No	Revision Note	Date	Signature	Checked
△				
△				



NOTES

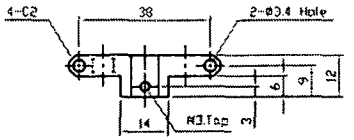
1. General Chamfer : C0.2 ~0.5
2. General Tolerance : ±0.1

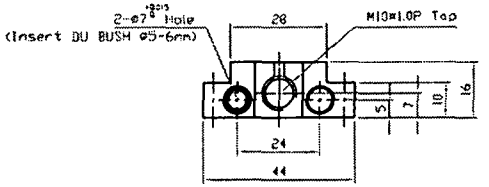
No	Drawing Name	Material	Q'ty	Finish
315	BRACKET - CYL.	A5052P 3t	1	WHITE ANODIZING

KIMM	Drawn	Check	Approv.	Title
	MWPARK 990901			FEEDER PART POTATO M/C

Unit	Scale	Finish Layout	Dwg.no	Sheet
mm	1/1	31.4 • 12 • 8	POT99 - 315	A3

Rev No	Revision Note	Date	Signature	Checked
△				
△				





NOTES

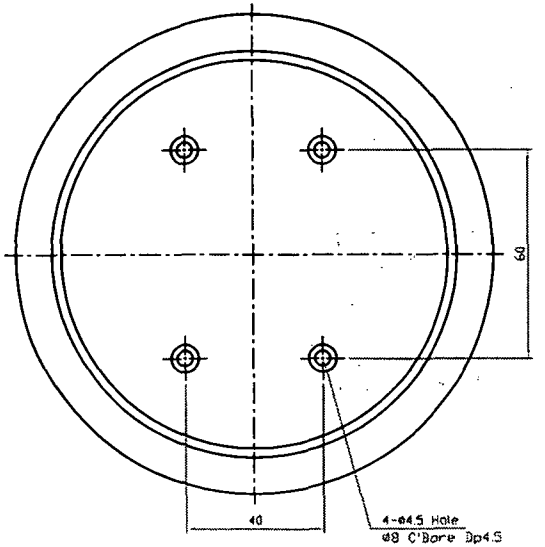
1. General Chamfer : C0.2 ~0.5
2. General Tolerance : ±0.1

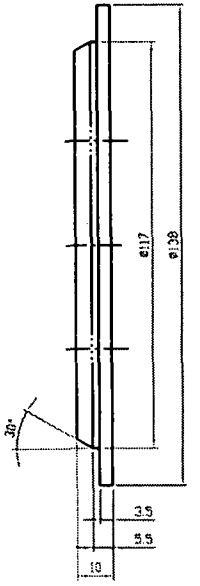
No	Drawing Name	Material	Q'ty	Finish
303	BRACKET	A6061-T6	4	WHITE ANODIZING

KIMM	Drawn	Check	Approv.	Title
	MWPARK 990901			FEEDER PART POTATO M/C

Unit	Scale	Finish Layout	Dwg.no	Sheet
mm	1/1	44 • 16 • 12	POT99 – 303	A3

Rev No	Revision Note	Date	Signature	Checked
△				
△				





NOTES

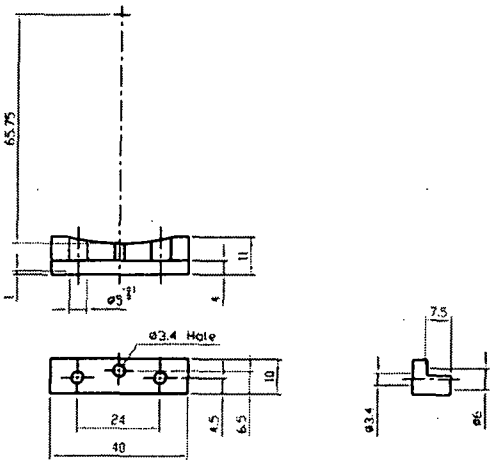
1. General Chamfer : C0.2 ~0.5
2. General Tolerance : ±0.1

No	Drawing Name	Material	Q'ty	Finish
309	BASE 2	MC NYLON 10t (WHITE)	2	-

KIMM	Drawn	Check	Approv.	Title
	MWPARK 990901			FEEDER PART POTATO M/C

Unit	Scale	Finish Layout	Dwg.no	Sheet
mm	1/1	ø138 * 10	POT99 – 309	A3

Rev No	Revision Note	Date	Signature	Checked
△				
△				



NOTES

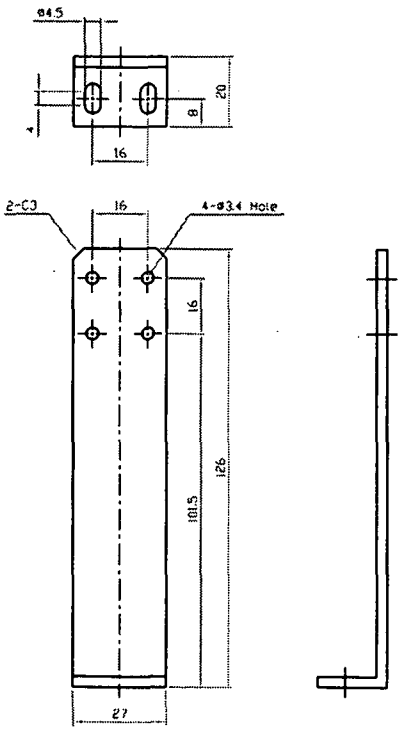
1. General Chamfer : C0.2 ~0.5
2. General Tolerance : ±0.1

No	Drawing Name	Material	Q'ty	Finish
302	PUSHER	MC NYLON (WHITE)	4	-

KIMM	Drawn	Check	Approv.	Title
	MWPARK 990901			FEEDER PART POTATO M/C

Unit	Scale	Finish Layout	Dwg.no	Sheet
mm	1/1	40 • 11 • 10	POT99 - 302	A3

Rev No	Revision Note	Date	Signature	Checked
△				
△				



t=3

NOTES

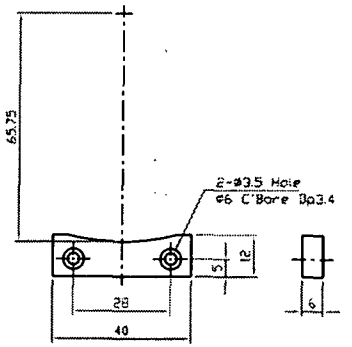
1. General Chamfer : C0.2 ~0.5
2. General Tolerance : ±0.1

No	Drawing Name	Material	Q'ty	Finish
307	BRACKET – CYL	SPC1 3t	4	Ni Plate

KIMM	Drawn	Check	Approv.	Title
	MWPARK 990901			FEEDER PART POTATO M/C

Unit	Scale	Finish Layout	Dwg.no		Sheet
mm	1/1	126 • 27 • 20	POT99 – 307		A3

Rev No	Revision Note	Date	Signature	Checked
△				
△				



NOTES

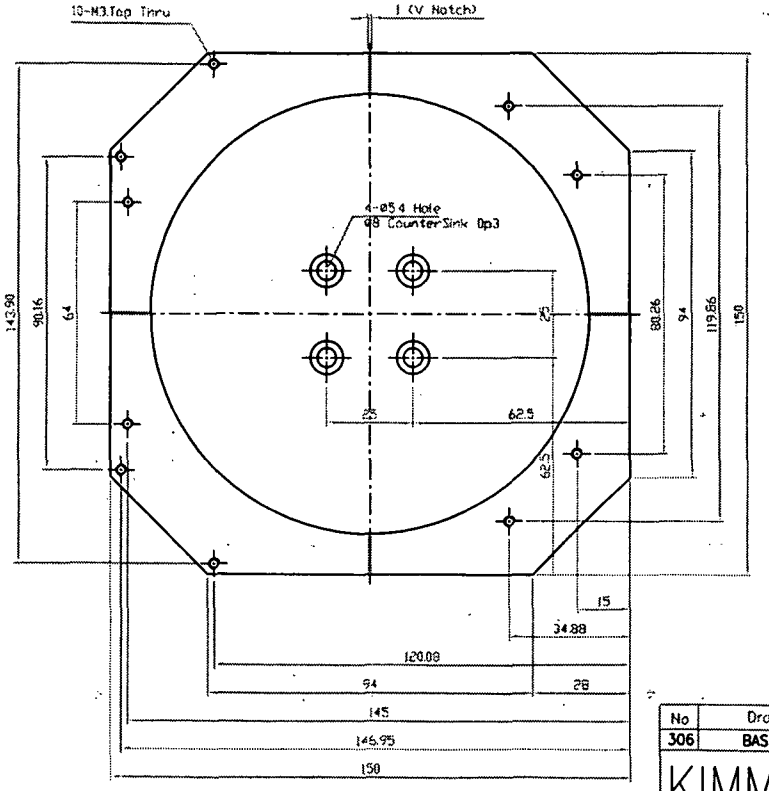
1. General Chamfer : C0.2 ~0.5
2. General Tolerance : ±0.1

No	Drawing Name	Material	Q'ty	Finish
301	STOPPER	HARD URETHANE	4	-

KIMM	Drawn	Check	Approv.	Title	FEEDER PART POTATO M/C
	MWPARK 990901				

Unit	Scale	Finish Layout	Dwg.no	Sheet
mm	1/1	40 • 12 • 6	POT99 - 301	A3

Rev No	Revision Note	Date	Signature	Checked
△				
△				



NOTES

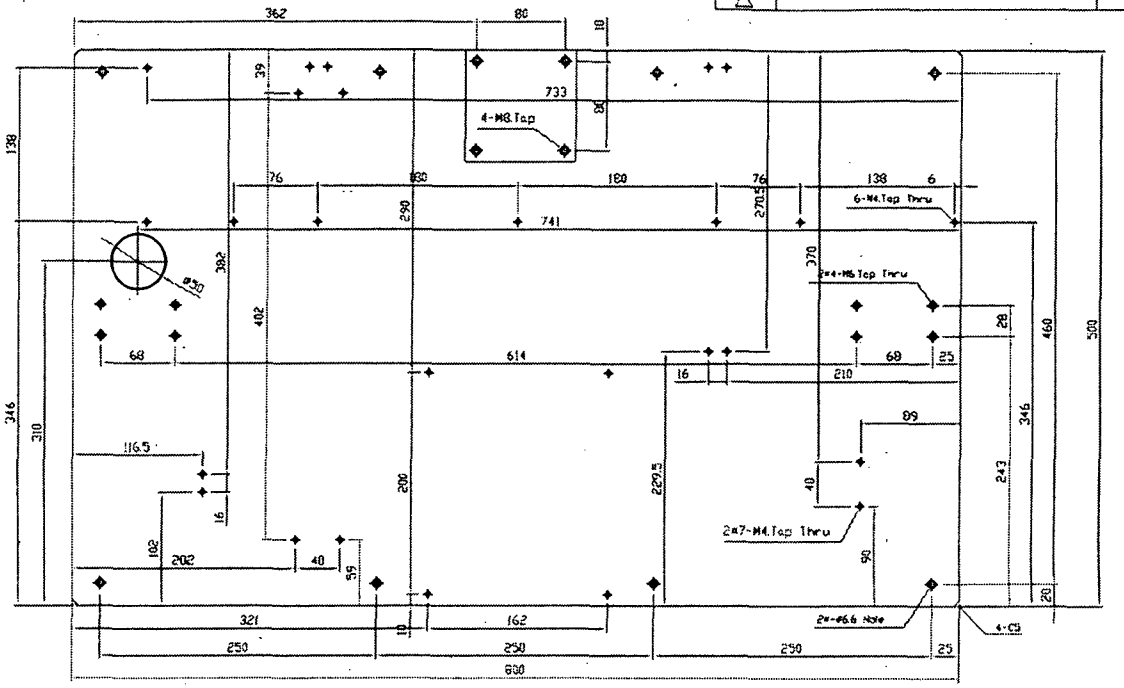
1. General Chamfer : C0.2 ~0.5
2. General Tolerance : ±0.1

No	Drawing Name	Material	Q'ty	Finish
306	BASE - SHALET	A5052P 8t	2	-

KIMM	Drawn	Check	Approv.	Title	FEEDER PART POTATO M/C
	MWPARK 990901				

Unit	Scale	Finish Layout	Dwg.no	Sheet
mm	1/1	150 * 150 * 8t	POT99 - 306	A3

Rev No	Revision Note	Date	Signature	Checked



NOTES

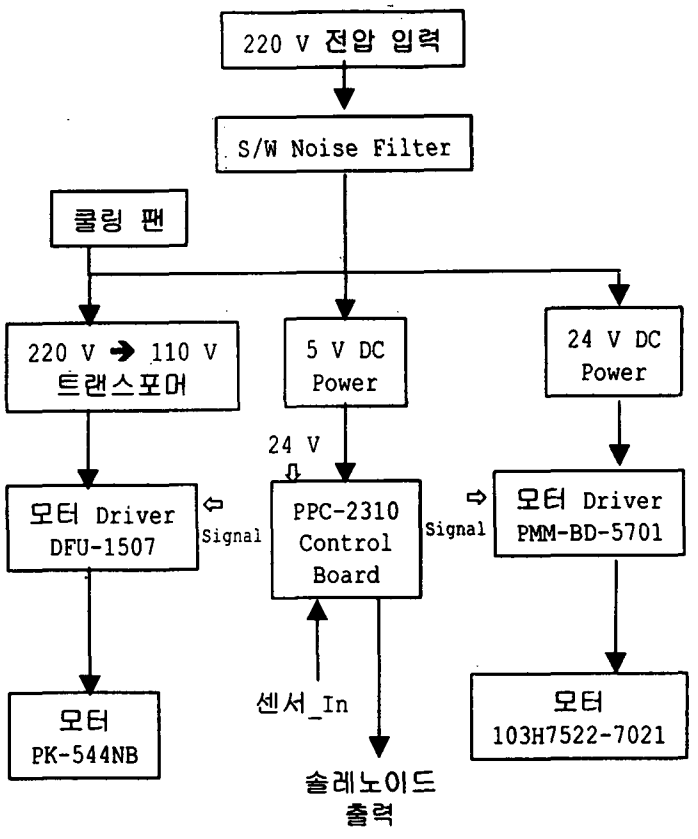
1. General Chamfer : C0.2 ~0.5
2. General Tolerance : ±0.1

No	Drawing Name	Material	Q'ty	Finish
310	PLATE - BASE	A5052P 12t	1	WHITE ANODIZING

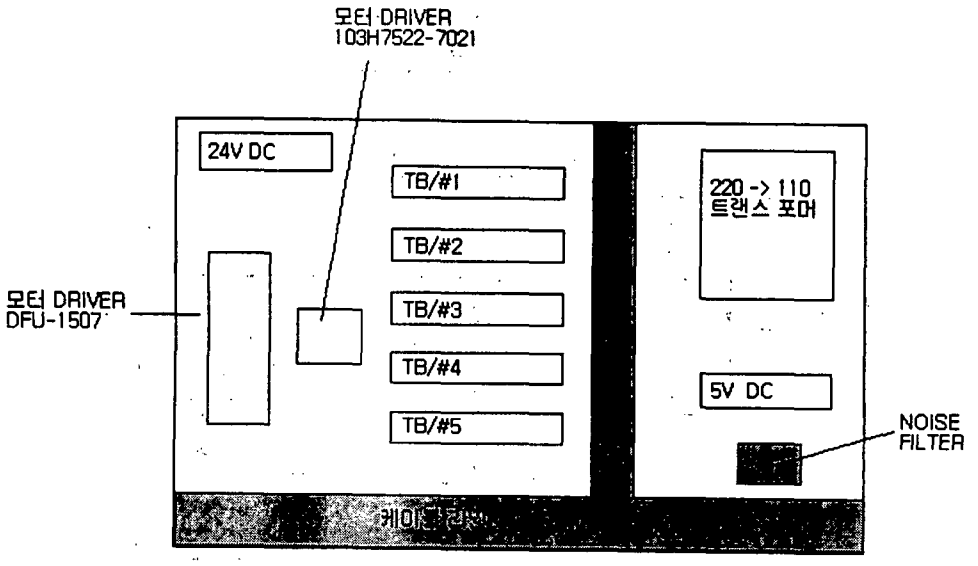
KIMM	Drawn	Check	Approv.	Title
	MWPARK 990901			FEEDER PART POTATO M/C

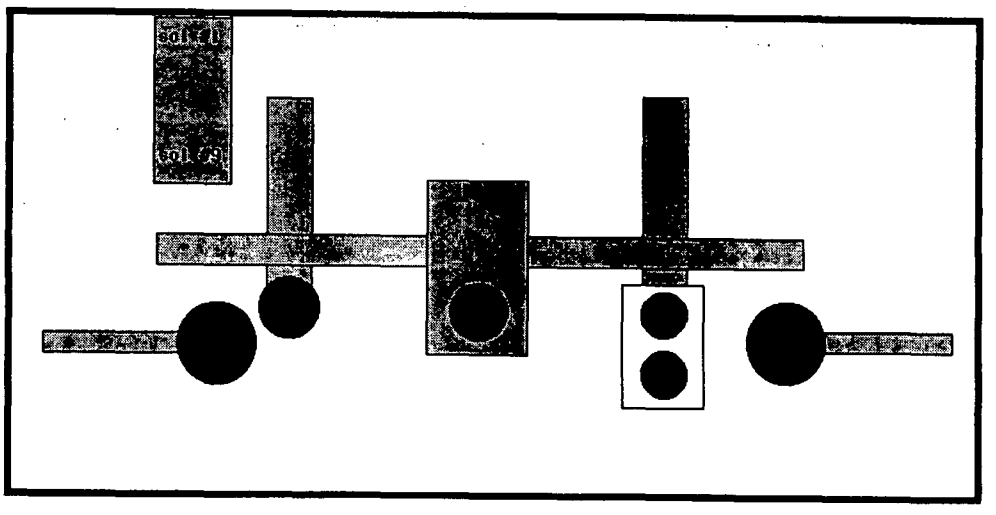
Unit	Scale	Finish Layout	Dwg.no	Sheet
mm	1/1	800 • 500 • 12t	POT99 - 310	A3

1. Driver Power Block Diagram.



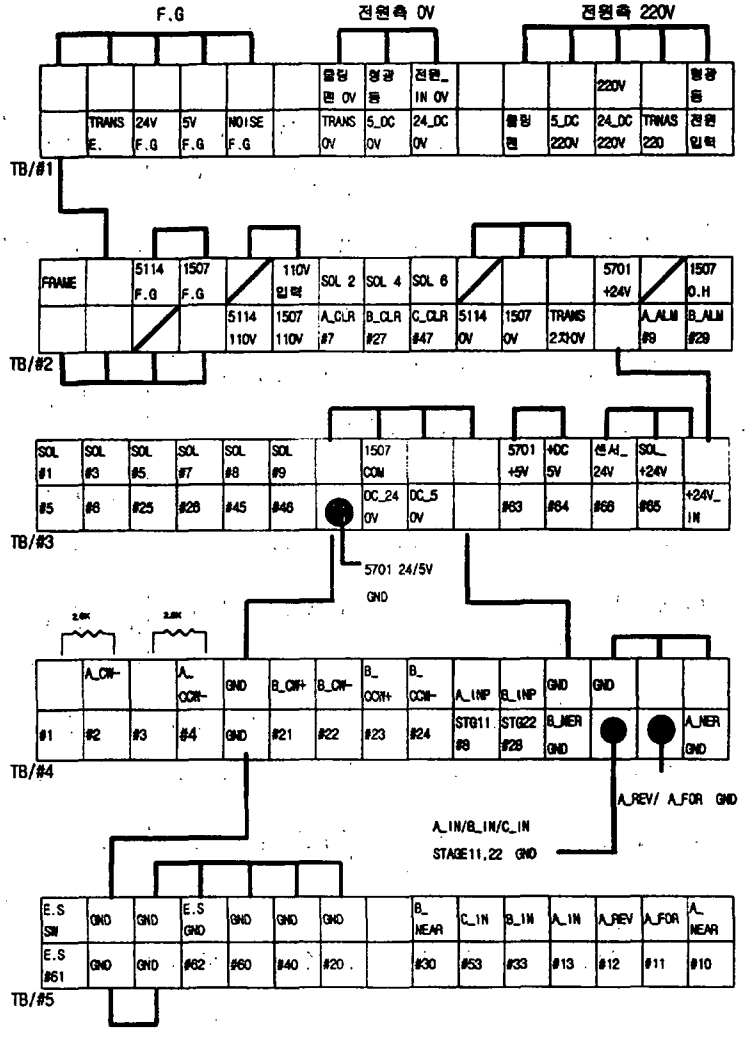
2. 기판 외형도





상층부 SOL의 배열도

3. 터미널 블록 배치도


F.G
전원측 0V
전원측 220V
TB/#1
TB/#2
TB/#3
TB/#4
TB/#5
5701 24/5V
GND
A_REV/ A_FOR GND
A_IN/B_IN/C_IN
STAGE11,22 GND

4. 센서 및 솔레노이드 DEFINE

4.1 센서부

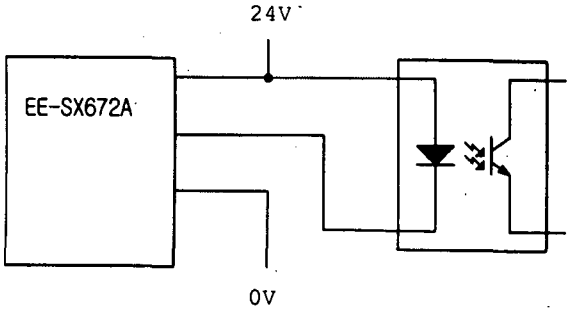
UNIT	신호명	센서명	BOARD ASSIGN	설명
X축 -원점 검출 유닛(L1)	XPOINT_IN1	EE - SX672A	PPC2310 (#12)	A_REV
X축 -원점 검출 유닛(L2)	XPOINT_IN2	EE - SX672A	PPC2310 (#10)	A_NEAR
X축 -원점 검출 유닛(R1)	XPOINT_R	EE - SX672A	PPC2310 (#11)	A_FOR
AIR CHUCK 원점 센서	ZPOINT_UP	EE - SX672A	PPC2310 (#30)	B_NEAR
AIR CYLINDER 원점 센서	RPOINT_IN	D-A93	PPC2310 (#13)	A_IN
공급STAGE #1 확인 센서	TPOINT_IN	W-13	PPC2310 (#33)	B_IN
공급STAGE #2 확인 센서	TPOINT_IN2	W-13	PPC2310 (#53)	C_IN
공급STAGE #11 확인 센서	TPOINT_IN11	W-13	PPC2310 (# 8)	A_INP
공급STAGE #22 확인 센서	TPOINT_IN22	W-13	PPC2310 (#28)	B_INP

4-2 솔레노이드 출력부

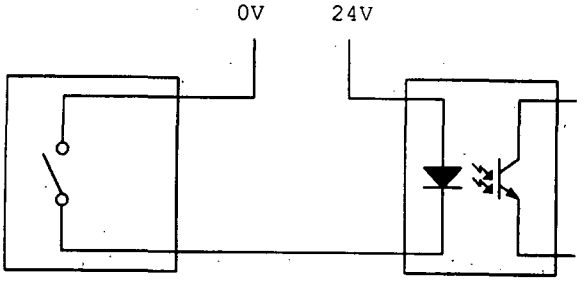
UNIT	동 작	신 호 명	BOARD ASSIGN	기타
Empty tray fixer	ON : 접시조임 OFF: 접시풀림	go_push	PPC2310 (#7)	A_CO
		un_push		
Empty tray FOR/REV	ON : 전방이송 OFF: 후방이송	go_front11	PPC2310 (#5)	A_CLR
		go_back11		
배양 Tray fixer	ON : 접시조임 OFF: 접시풀림	go_push2	PPC2310 (#6)	A_SERVO
		un_push2		
배양 Tray FOR/REV	ON : Tray 이송 OFF: Tray 후송	go_front22	PPC2310 (#27)	B_CLR
		go_back22		
COVER BASE	ON : 전방이송 OFF: 후방이송	go_front2	PPC2310 (#25)	B_CO
		go_back2		
비젼 Stage rotary	ON : 90° 회전 OFF: 정상상태	go_rot2	PPC2310 (#47)	C_CLR
		go_org2		
Tray 배출	ON : Tray이송 OFF: Tray후송	go_front1	PPC2310 (#26)	B_SERVO
		go_back1		
Air chuck	ON : Arm 조임 OFF: Arm 풀림	go_arm	PPC2310 (#45)	C_CO
		un_arm		
Rotary cylinder	ON : 90° 회전 OFF: 정상상태	go_rot	PPC2310 (#46)	C_SERVO
		go_org		

5. 센서/SOLENOID 동작 원리도.

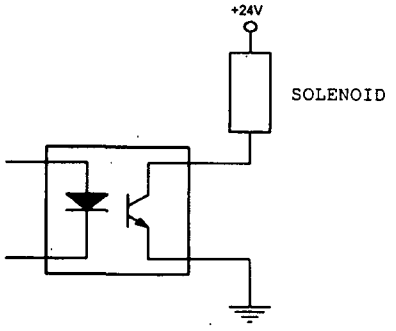
EE - SX672A



w-13 / D-A93



SOLENOID



6. 전력 소모량

모터 드라이버(DFU1507) : 110V/1.1A = 실효치 85.5W

모터 드라이버(PMM-BD-5701) : DC24V/2.5A + DC5V/0.15A = 67.5W

센서 EE-SX672 : 24V/8.8mA (4개) = 0.9W

센서 W-13 : 24V/8.8mA (4개) = 0.8W

센서 D-A93 : 24V/8.8mA = 0.2W

솔레노이드 : 24V/21mA (6개) = 3W

쿨링 팬 : 220/0.07 = 실효치 10.8W

4.7W

TOTAL : 168.5W

사용된 부품 일람표

트랜스 포머 : 300VA (220 ->110 운영 社)

POWER : 24V / 3.5A (FINE.SUNTRONICS)

5V / 3A (FINE.SUNTRONICS)

드라이버 : DFU-1507 (ORIENTAL 社 5相 STEP모터)

PMM-BD-5701 (SANYO 社 5相 STEP모터)

모터 : PK-544NB (ORIENTAL)

103H7522-7021 (SANYO)

솔레노이드 : SY3160-5L-C4 (SMC 社)

센서 : EE-SX672 (OMRON 社)

W-13 (TPC 社)

D-A93 (SMC 社)

터미널 블럭 15PIN

저항 : 2.6K

여 백

2. 시스템 관련 S/W Source

여 백

```
    1: // vt52.cpp : Defines the class behaviors for the application.
    2: //
    3:
    4: #include "stdafx.h"
    5: #include "vt52.h"
    6: #include "mot.h"
    7: #include "data.h"
    8:
    9: #include "MainFrm.h"
   10: #include "vt52Doc.h"
   11: #include "vt52View.h"
   12: #include "vision.h"
   13: #include "robot.h"
   14: #include "global.h"
   15: #include "head.h"
   16:
   17: // instance handle supplied by winmain.c
   18: HINSTANCE sampleHInstance;
   19: LRESULT CALLBACK VideoWindowProc(HWND hWnd, unsigned msg, WPARAM wParam,
   20:                                  LPARAM lParam);
   21: void UpdateRect (HWND hWnd, int xPos, int yPos);
   22: void RECT draw  (HDC dc, RECT *ROI);
   23: c UInt32 CmdInterpret ( LPVOID );
   24:
   25: extern int  CAMERA FLAG;
   26: /////////////////////////////////////////////////////////////////////////////
   27: // CVt52App
   28:
   29: BEGIN MESSAGE MAP(CVt52App, CWinApp)
   30:     //{{AFX MSG MAP(CVt52App)
   31:     ON COMMAND(ID APP ABOUT, OnAppAbout)
   32:     ON COMMAND(IDD CCIR, OnCcir)
   33:     ON COMMAND(IDD NTSC, OnNtsc)
   34:     ON COMMAND(IDD PAL, OnPal)
   35:     ON COMMAND(IDD RS170, OnRs170)
   36:     ON COMMAND(IDD ONE, OnOne)
   37:     ON COMMAND(IDD THIRD, OnThird)
   38:     ON COMMAND(IDD SECOND, OnSecond)
   39:     ON COMMAND(IDD FOUR, OnFour)
   40:     ON COMMAND(ID IMAGE READ, OnImageRead)
   41:     ON COMMAND(ID IMAGE WRITE, OnImageWrite)
   42:     ON COMMAND(IDD IMAGEGRAB, OnImagegrab)
   43:     ON UPDATE COMMAND UI(IDD IMAGEGRAB, OnUpdateImagegrab)
   44:     ON UPDATE COMMAND UI(ID IMAGE WRITE, OnUpdateImageWrite)
   45:     ON COMMAND(ID RS232C, OnRs232c)
   46:     ON UPDATE COMMAND UI(IDD START, OnUpdateStart)
   47:     ON COMMAND(IDD STOP, OnStop)
   48:     ON UPDATE COMMAND UI(IDD STOP, OnUpdateStop)
   49:     ON UPDATE COMMAND UI(ID RS232C, OnUpdateRs232c)
   50:     ON COMMAND(IDD START, OnStart)
   51:     ON COMMAND(IDD AUTO, OnAuto)
   52:     ON UPDATE COMMAND UI(IDD AUTO, OnUpdateAuto)
   53:     ON COMMAND(ID ROBOT, OnRobot)
   54:     ON COMMAND(IDD SUPPLY FULLTRAY, OnSupplyFulltray)
   55:     ON COMMAND(IDD SUPPLY EMPTYTRAY, OnSupplyEmptytray)
   56:     ON COMMAND(IDD OUTPUT CUTTRAY, OnOutputCuttray)
   57:     ON COMMAND(IDD OUTPUT INSTRAY, OnOutputInstray)
   58:     ON UPDATE COMMAND UI(IDD OUTPUT CUTTRAY, OnUpdateOutputCuttray)
   59:     ON UPDATE COMMAND UI(IDD OUTPUT INSTRAY, OnUpdateOutputInstray)
   60:     ON UPDATE COMMAND UI(IDD SUPPLY EMPTYTRAY, OnUpdateSupplyEmptytray)
   61:     ON UPDATE COMMAND UI(IDD SUPPLY FULLTRAY, OnUpdateSupplyFulltray)
   62:     ON UPDATE COMMAND UI(ID TRAYPOS SETUP, OnUpdateTrayposSetup)
   63:     //}}AFX MSG MAP
   64:     // Standard file based document commands
   65:     ON COMMAND(ID FILE NEW, CWinApp::OnFileNew)
   66:     ON COMMAND(ID FILE OPEN, CWinApp::OnFileOpen)
   67:     // Standard print setup command
   68:     ON COMMAND(ID FILE PRINT SETUP, CWinApp::OnFilePrintSetup)
   69: END MESSAGE MAP()
   70:
   71: /////////////////////////////////////////////////////////////////////////////
   72: // CVt52App construction
   73:
   74: CVt52App::CVt52App ()
   75: {
```

```
   76:     // TODO: add construction code here,
   77:     // Place all significant initialization in InitInstance
   78:     // Enable/Disable the main Menu Bar
   79:     Camera = 1;
   80:     SelectDone = FALSE;
   81:     ModeSelect = FALSE;
   82:     snapdone   = FALSE;
   83:     RS232Done  = FALSE;
   84:     AutoDone   = FALSE;
   85:     AUTORUN    = FALSE;
   86:     HOMEDONE   = FALSE;
   87:
   88:     // Initialize the rs232 Port
   89:     mrs232c.SetComPort ( 2, 9600, 8, 0, 0 );
   90:     mrs232c.CreateCommInfo ();
   91:     mrs232c.OpenComPort ();
   92: }
   93:
   94: CVt52App::~CVt52App ()
   95: {
   96:     // Unallocate the resource of RS232 Port
   97:     mrs232c.DestroyComm ();
   98: }
   99: /////////////////////////////////////////////////////////////////////////////
  100: // The one and only CVt52App object
  101:
  102: CVt52App theApp;
  103:
  104: /////////////////////////////////////////////////////////////////////////////
  105: // CVt52App initialization
  106:
  107: BOOL CVt52App::InitInstance()
  108: {
  109:     AfxEnableControlContainer();
  110:
  111:     // Standard initialization
  112:     // If you are not using these features and wish to reduce the size
  113:     //  of your final executable, you should remove from the following
  114:     //  the specific initialization routines you do not need.
  115:
  116: #ifdef  AFXDLL
  117:     Enable3dControls();         // Call this when using MFC in a shared DLL
  118: #else
  119:     Enable3dControlsStatic();  // Call this when linking to MFC statically
  120: #endif
  121:
  122:     // Change the registry key under which our settings are stored.
  123:     // You should modify this string to be something appropriate
  124:     // such as the name of your company or organization.
  125:     SetRegistryKey( T("Local AppWizard-Generated Applications"));
  126:
  127:     LoadStdProfileSettings();  // Load standard INI file options (including
       MRU)
  128:
  129:     // Register the application's document templates.  Document templates
  130:     //  serve as the connection between documents, frame windows and views.
  131:
  132:     CSingleDocTemplate* pDocTemplate;
  133:     pDocTemplate = new CSingleDocTemplate(
  134:         IDR MAINFRAME,
  135:         RUNTIME CLASS(CVt52Doc),
  136:         RUNTIME CLASS(CMainFrame),       // main SDI frame window
  137:         RUNTIME CLASS(CVt52View));
  138:     AddDocTemplate(pDocTemplate);
  139:
  140:     // Parse command line for standard shell commands, DDE, file open
  141:     CCommandLineInfo cmdInfo;
  142:     ParseCommandLine(cmdInfo);
  143:
  144:     // Dispatch commands specified on the command line
  145:     if (!ProcessShellCommand(cmdInfo))
  146:         return FALSE;
  147:
  148:     // The one and only window has been initialized, so show and update it.
  149:     m pMainWnd->ShowWindow(SW SHOW);
```

```
150:     m pMainWnd->UpdateWindow();
151:
152:     //CWnd *pWnd = AfxGetMainWnd ();
153:     //CMenu *pMenu = pWnd->GetMenu ();
154:     CMenu *pMenu = m pMainWnd->GetMenu ();
155:
156:     pMenu->EnableMenuItem ( MENUCALIB,    MF BYPOSITION | MF DISABLED | MF GRAYE
         D );
157:     pMenu->EnableMenuItem ( MENUTOOL,     MF BYPOSITION | MF DISABLED | MF GRAYE
         D );
158:     pMenu->EnableMenuItem ( MENUINSPECT, MF BYPOSITION | MF DISABLED | MF GRAYE
         D );
159: // pMenu->EnableMenuItem ( MENURUN,      MF_BYPOSITION | MF_DISABLED | MF_GRAYE
         D );
160:     pMenu->EnableMenuItem ( MENURUN,      MF BYPOSITION | MF ENABLED );
161:     pMenu->EnableMenuItem ( MENUCOMM,     MF BYPOSITION | MF ENABLED );
162:
163:     return TRUE;
164: }
165:
166: /////////////////////////////////////////////////////////////////////////////
167: // CAboutDlg dialog used for App About
168:
169: class CAboutDlg : public CDialog
170: {
171: public:
172:     CAboutDlg();
173:
174: // Dialog Data
175:     //{{AFX_DATA(CAboutDlg)
176:     enum { IDD = IDD_ABOUTBOX };
177:     //}}AFX_DATA
178:
179:     // ClassWizard generated virtual function overrides
180:     //{{AFX_VIRTUAL(CAboutDlg)
181:     protected:
182:     virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
183:     //}}AFX_VIRTUAL
184:
185: // Implementation
186: protected:
187:     //{{AFX_MSG(CAboutDlg)
188:         // No message handlers
189:     //}}AFX_MSG
190:     DECLARE MESSAGE MAP()
191: };
192:
193: CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
194: {
195:     //{{AFX_DATA_INIT(CAboutDlg)
196:     //}}AFX_DATA_INIT
197: }
198:
199: void CAboutDlg::DoDataExchange(CDataExchange* pDX)
200: {
201:     CDialog::DoDataExchange(pDX);
202:     //{{AFX_DATA_MAP(CAboutDlg)
203:     //}}AFX_DATA_MAP
204: }
205:
206: BEGIN MESSAGE MAP(CAboutDlg, CDialog)
207:     //{{AFX_MSG_MAP(CAboutDlg)
208:         // No message handlers
209:     //}}AFX_MSG_MAP
210: END MESSAGE MAP()
211:
212: // App command to run the dialog
213: void CVt52App::OnAppAbout()
214: {
215:     CAboutDlg aboutDlg;
216:     aboutDlg.DoModal();
217: }
218:
219: /////////////////////////////////////////////////////////////////////////////
220: // CVt52App commands
```

```
221:
222: void CVt52App::MenuControl ( int menu, int status )
223: {
224:    CWnd *pWnd = AfxGetMainWnd ();
225:    CMenu *pMenu = pWnd->GetMenu ();
226:    if ( status == ENABLE )
227:       pMenu->EnableMenuItem ( menu, MF BYPOSITION | MF ENABLED );
228:    else
229:       pMenu->EnableMenuItem ( menu, MF BYPOSITION | MF DISABLED | MF GRAYED );
230:
231:    ::DrawMenuBar ( pWnd->m hWnd );
232: }
233: void CVt52App::OnCcir()
234: {
235:    // TODO: Add your command handler code here
236:    video mode = CACQ CCIR;
237:
238:    VIDEO init();
239:    CAMERA FLAG = TRUE;
240: }
241:
242: void CVt52App::OnNtsc()
243: {
244:    // TODO: Add your command handler code here
245:    video mode = CACQ RGB RS170;
246:
247:    VIDEO init();
248:    CAMERA FLAG = TRUE;
249: }
250:
251: void CVt52App::OnPal()
252: {
253:    // TODO: Add your command handler code here
254:    video mode = CACQ PAL;
255:
256:    VIDEO init();
257:    CAMERA FLAG = TRUE;
258: }
259:
260: void CVt52App::OnRs170()
261: {
262:    // TODO: Add your command handler code here
263:    video mode = CACQ RS170;
264:
265:    VIDEO init();
266:    CAMERA FLAG = TRUE;
267: }
268:
269: void CVt52App::OnOne()
270: {
271:    // TODO: Add your command handler code here
272:    CamSelect (1);
273: }
274:
275: void CVt52App::OnSecond()
276: {
277:    // TODO: Add your command handler code here
278:    CamSelect (2);
279: }
280:
281: void CVt52App::OnThird()
282: {
283:    // TODO: Add your command handler code here
284:    CamSelect (3);
285: }
286:
287: void CVt52App::OnFour()
288: {
289:    // TODO: Add your command handler code here
290:    CamSelect (4);
291: }
292:
293: CACQ resource oh oh Resource;
294: CACQ channel oh  oh Channel;
295: CVID oh  oh Video;
```

```
296: CPB oh   windowHandle[MAX FRAME];
297: CIC oh  oh Console[MAX FRAME];
298: int   DISPLAYCONSOLE;
299:
300: void CVt52App::VIDEO_init()
301: {
302:    //RECT     vgaRect;
303:    HWND videoWindow;
304:    CACQ rect maskRect;
305:
306:    int size[8][2] =
307:    { { 640, 480}, { 768, 754}, {640, 480}, { 720, 576},
308:    { 512, 480}, { 720, 546}, { 640, 480}, { 768, 574} };
309:
310:    // Check the Vilid No. of Camera
311:    if (Camera < 1 || Camera > 6) return;
312:
313:    // Check the Valid Video Mode
314:    // initialize the resource and setup the Channel
315:    #ifdef VP54
316:       CVP54 resource default setup (&vp54 resource);
317:       if ( video mode == CACQ RGB RS170 || video mode == CACQ_PAL )
318:          vp54 resource.format = video mode | CACQ PLANE RED;
319:       // vp54 resource.format = video mode | CACQ_PLANE_GREEN;
320:       else vp54_resource.format = video_mode;
321:
322:       vp54 resource.max area.x = 0;
323:       vp54 resource.max area.y = 0;
324:       vp54 resource.max area.w = size[video mode][0];
325:       vp54 resource.max area.h = size[video mode][1];
326:       vp54 resource.buffers = 4;
327:
328:       oh Resource = CVP54 resource alloc setup (&vp54 resource);
329:       if (oh Resource == ckAcqResource BadValue )
330:          // || oh_Resource == ck_VP54_HardwareNotInitialized)
331:          return;
332:       oh Channel = CVP54 channel alloc (oh Resource, Camera);
333:       if (oh Channel == ckAcqChannel BadValue ||
334:          oh Channel == ckBadObjectType)
335:          return;
336:    #else
337:       CVP52 resource default setup (&vp52 resource);
338:       if ( video mode == CACQ RGB RS170 || video mode == CACQ_PAL )
339:          vp52 resource.format = video mode | CACQ PLANE RED;
340:       // vp52 resource.format = video mode | CACQ_PLANE_GREEN;
341:       else vp52_resource.format = video_mode;
342:
343:       vp52 resource.max area.x = 0;
344:       vp52 resource.max area.y = 0;
345:       vp52 resource.max area.w = size[video mode][0];
346:       vp52 resource.max area.h = size[video mode][1];
347:       vp52 resource.buffers = 4;
348:
349:       oh Resource = CVP52 resource alloc setup (&vp52 resource);
350:       if (oh Resource == ckAcqResource BadValue )
351:          // || oh_Resource == ck_VP52_HardwareNotInitialized)
352:          return;
353:       oh Channel = CVP52 channel alloc (oh Resource, Camera);
354:       if (oh Channel == ckAcqChannel BadValue ||
355:          oh Channel == ckBadObjectType)
356:          return;
357:    #endif
358:
359:    oh Video = CVID alloc (oh Resource);
360:    if (oh Video == ckBadObjectType ||
361:       oh Video == ckVideoMixer ResourceInUse)
362:       return;
363:
364:    // Determine the actual size and location of the region in VGA Screen
365:    vgaWindow.x = 20;
366:    vgaWindow.y = 90;
367:    vgaWindow.w = size[video mode][0];
368:    vgaWindow.h = size[video mode][1];
369:    // Create window to hold video.
370:     // Register window class
```

```
371:     wc.style = 0;
372:     wc.lpfnWndProc = VideoWindowProc;
373:     wc.cbClsExtra = 0;
374:     wc.cbWndExtra = 0;
375:     sampleHInstance = AfxGetInstanceHandle();
376:     wc.hInstance = sampleHInstance;
377:     wc.hIcon = NULL;
378:     wc.hCursor = NULL;
379:     wc.hbrBackground = NULL;
380:     wc.lpszMenuName =  NULL;
381:     wc.lpszClassName = "VideoWindowClass";
382:     if (!RegisterClass(&wc)) return;
383:
384:     // Compute starting origin and size for CreateWindow call, so created
385:     // window has the appropriate client area.
386:     vgaRect.left    = (int) vgaWindow.x;
387:     vgaRect.right   = (int)(vgaWindow.x + vgaWindow.w);
388:     vgaRect.top     = (int) vgaWindow.y;
389:     vgaRect.bottom  = (int)(vgaWindow.y + vgaWindow.h);
390:     AdjustWindowRect(&vgaRect, WS OVERLAPPED | WS CAPTION | WS SYSMENU, 0);
391:
392:     // Create and show window.
393:     videoWindow = CreateWindow("VideoWindowClass", "Live Video",
394:                         WS OVERLAPPED | WS CAPTION | WS SYSMENU,
395:                            vgaRect.left, vgaRect.top, vgaRect.right -
       vgaRect.left,
396:                            vgaRect.bottom - vgaRect.top,
397:                            NULL, NULL, sampleHInstance, NULL);
398:     ShowWindow (videoWindow, SW SHOWNORMAL);
399:
400:     // Register the video window and begin display of video.
401:     #ifdef VP54
402:        CVP54 register video window (oh Resource, oh Video, (int)videoWindow);
403:     #else
404:        CVP52 register video window (oh Resource, oh Video, (int)videoWindow);
405:     #endif
406:
407:     // set the mask area in video screen
408:     maskRect.x  = 100;
409:     maskRect.y = 100;
410:     maskRect.w  = 200;
411:     maskRect.h  = 200;
412:
413:     #ifdef VP54
414:        //CVP54 add mask rect (oh Resource, oh Video, &maskRect);
415:        //CVP54_flush_mask (oh_Resource, oh_Video);
416:     #else
417:        //CVP52 add mask rect (oh Resource, oh Video, &maskRect);
418:        //CVP52_flush_mask (oh_Resource, oh_Video);
419:     #endif
420:     CVID begin display(oh Video, &vgaWindow);
421:     DISPLAYCONSOLE = 0;
422:     if ( ModeSelect == FALSE )
423:     {
424:        char buf [50];
425:        int nIDCheckItem;
426:        CWnd *pWnd = AfxGetMainWnd ();
427:        CMenu *pMenu = pWnd->GetMenu ();
428:        CMenu *pSub1Menu = pMenu->GetSubMenu ( MENUCAMERA );
429:        HMENU hSub2Menu = ::GetSubMenu ( pSub1Menu->m hMenu, MENUSUB1MODE );
430:        if ( hSub2Menu != NULL )
431:        {
432:           int ret;
433:
434:           if ( video mode == CACQ RGB RS170 )
435:              nIDCheckItem = MENUSUB2RGB;
436:           else if ( video mode == CACQ PAL )
437:              nIDCheckItem = MENUSUB2PAL;
438:           else if ( video mode == CACQ RS170 )
439:              nIDCheckItem = MENUSUB2RS170;
440:           else if ( video mode == CACQ CCIR )
441:              nIDCheckItem = MENUSUB2CCIR;
442:
443:           ret = ::GetMenuString ( hSub2Menu, nIDCheckItem, buf, 15,
       MF BYPOSITION );
```

```
444:            ret = ::CheckMenuItem ( hSub2Menu, nIDCheckItem, MF BYPOSITION |
    MF CHECKED );
445:         }
446:      }
447:
448:      SelectDone = TRUE;
449:      ModeSelect = TRUE;
450:      if ( RS232Done == TRUE && AUTORUN == TRUE )
451:         MenuControl ( MENURUN, ENABLE );
452: }
453:
454: void CVt52App::CamSelect ( int NewCamera )
455: {
456:    if (Camera != NewCamera)
457:    {
458:       if ( ModeSelect )
459:       {
460:          CVID end display  (oh Video);
461:          CACQ channel free (oh Channel);
462:    #ifdef VP54
463:          oh Channel =  CVP54 channel alloc (oh Resource, NewCamera);
464:    #else
465:          oh Channel =  CVP52 channel alloc (oh Resource, NewCamera);
466:    #endif
467:          CVID live camera set (oh Video, NewCamera);
468:          CVID begin display   (oh Video, &vgaWindow);
469:    #ifdef VP54
470:          // CVP54 clear all mask rects(gResource, gVideoArea); // added1214
471:          // CVP54_flush_mask(gResource, gVideoArea); // added1214
472:    #else
473:          // CVP54 clear all mask rects(gResource, gVideoArea); // added1214
474:          // CVP54_flush_mask(gResource, gVideoArea); // added1214
475:    #endif
476:          // PMAINFRAMEWND->Refresh_Activated_Window();
477:       }
478:    }
479:
480:    CWnd *pWnd = AfxGetMainWnd ();
481:    CMenu *pMenu = pWnd->GetMenu ();
482:    CMenu *pSub1Menu = pMenu->GetSubMenu ( MENUCAMERA );
483:    HMENU hSub2Menu = ::GetSubMenu ( pSub1Menu->m hMenu, MENUSUB1CAMNO );
484:    if ( hSub2Menu != NULL )
485:    {
486:       int ret;
487:
488:       if ( Camera != NewCamera )
489:          ret = ::CheckMenuItem ( hSub2Menu, Camera-1, MF BYPOSITION |
    MF UNCHECKED );
490:       ret = ::CheckMenuItem ( hSub2Menu, NewCamera-1, MF BYPOSITION |
    MF CHECKED );
491:
492:       ::DrawMenuBar ( pWnd->m hWnd );
493:    }
494:
495:    // Update Camera No.
496:    Camera = NewCamera;
497: }
498:
499: void OnWindowPosChanging(HWND hWnd, LPWINDOWPOS infoP)
500: {
501:   /*  Restrict window movement and sizing to that supported by
502:       the video area, using the CVID_negotiate function. */
503:    RECT clientRect, windowRect;
504:    RECT clientOffset;
505:    CACQ rect vidRect;
506:    GetClientRect(hWnd, &clientRect);
507:    GetWindowRect(hWnd, &windowRect);
508:    MapWindowPoints(hWnd, 0, (LPPOINT)&clientRect, 2);
509:    clientOffset.left   = clientRect.left   - windowRect.left;
510:    clientOffset.top    = clientRect.top    - windowRect.top;
511:    clientOffset.right  = clientRect.right  - windowRect.right;
512:    clientOffset.bottom = clientRect.bottom - windowRect.bottom;
513:    if (infoP->flags & SWP_NOMOVE)
514:    {
515:       infoP->x = windowRect.left;
```

```
516:        infoP->y = windowRect.top;
517:     }
518:     if (infoP->flags & SWP_NOSIZE)
519:     {
520:        infoP->cx = windowRect.right - windowRect.left;
521:        infoP->cy = windowRect.bottom - windowRect.top;
522:     }
523:     vidRect.x = infoP->x + clientOffset.left;
524:     vidRect.y = infoP->y + clientOffset.top;
525:     vidRect.w = infoP->cx - clientOffset.left + clientOffset.right;
526:     vidRect.h = infoP->cy - clientOffset.top  + clientOffset.bottom;
527:     infoP->x  = vidRect.x - clientOffset.left;
528:     infoP->y  = vidRect.y - clientOffset.top;
529:     infoP->cx = vidRect.w + clientOffset.left - clientOffset.right;
530:     infoP->cy = vidRect.h + clientOffset.top - clientOffset.bottom;
531: }
532:
533:
534: RECT gClientAreaOffsets, ROIRect;
535:
536: LRESULT CALLBACK VideoWindowProc(HWND hWnd, unsigned msg, WPARAM wParam,
537:                                  LPARAM lParam)
538: {
539:   PAINTSTRUCT ps;
540:   HPEN oldPen;
541:   HPEN pen;
542:   HDC dc;
543:   CVt52App *PApp = (CVt52App *) AfxGetApp ();
544:
545:   switch(msg)
546:   {
547:     case WM_CREATE:
548:     {
549:       RECT client, windowRect;
550:       GetWindowRect(hWnd, &windowRect);
551:       GetClientRect(hWnd, &client);
552:       MapWindowPoints(hWnd, 0, (LPPOINT)&client, 2);
553:       gClientAreaOffsets.left = client.left - windowRect.left;
554:       gClientAreaOffsets.right = windowRect.right - client.right;
555:       gClientAreaOffsets.top = client.top - windowRect.top;
556:       gClientAreaOffsets.bottom = windowRect.bottom - client.bottom;
557:       break;
558:     }
559:
560:     case WM_PAINT:
561:     {
562:        pen = CreatePen(PS_SOLID, 1, RGB(255,0,0));
563:       BeginPaint(hWnd, &ps);
564:       EndPaint(hWnd, &ps);
565:
566:       dc = GetDC(hWnd);
567:       oldPen = (HPEN) SelectObject(dc, pen);
568:       SelectObject(dc, oldPen);
569:
570:       ReleaseDC(hWnd, dc);
571:       DeleteObject(pen);
572:       break;
573:     }
574:
575:     case WM_WINDOWPOSCHANGING:
576:     {
577:       /*
578:         Get the desired window position from the infoP structure.
579:         The infoP gives the desired position and size of the *entire*
580:         window, not just its client area.
581:         Compute the corresponding client area of the window.
582:         Negotiate a valid position and update infoP to move to the valid
583:         position.
584:       */
585:     // OnWindowPosChanging(hWnd, (LPWINDOWPOS) lParam);
586:     // return DefWindowProc(hWnd, msg, wParam, lParam);
587:
588:       LPWINDOWPOS infoP = (LPWINDOWPOS) lParam;
589:       RECT rect;
590:       CACQ rect vidRect;
```

```
591:        GetClientRect(hWnd, &rect);
592:        MapWindowPoints(hWnd, 0, (LPPOINT)&rect, 2);
593:        if (!(infoP->flags & SWP_NOMOVE))
594:        {
595:     rect.left = infoP->x + gClientAreaOffsets.left;
596:     rect.top = infoP->y + gClientAreaOffsets.top;
597:        }
598:        if (!(infoP->flags & SWP_NOSIZE))
599:        {
600:     rect.right = rect.left + infoP->cx - gClientAreaOffsets.left -
601:       gClientAreaOffsets.right;
602:     rect.bottom = rect.top + infoP->cy - gClientAreaOffsets.top -
603:       gClientAreaOffsets.bottom;
604:        }
605:        vidRect.x = rect.left;
606:        vidRect.y = rect.top;
607:        vidRect.w = rect.right - rect.left;
608:        vidRect.h = rect.bottom - rect.top;
609:       // CVID window negotiate(oh Video, &vidRect);
610:        infoP->x = vidRect.x - gClientAreaOffsets.left;
611:        infoP->y = vidRect.y - gClientAreaOffsets.top;
612:        infoP->cx = vidRect.w + gClientAreaOffsets.left +
613:     gClientAreaOffsets.right;
614:        infoP->cy = vidRect.h + gClientAreaOffsets.top +
615:     gClientAreaOffsets.bottom;
616:        return DefWindowProc(hWnd, msg, wParam, lParam);
617:     }
618:
619:     // if the input point of mouse is not equal to the point of rectangle,
620:     // retry to pick the correct point.
621:     case WM_LBUTTONDOWN:
622:     {
623:        int xPos,yPos;
624:        xPos = LOWORD (lParam);
625:        yPos = HIWORD (lParam);
626:
627:        if (xPos >= (ROIRect.left - 4) && xPos <= (ROIRect.left + 4)
628:         && yPos >= (ROIRect.top - 4) && yPos <= (ROIRect.top + 4) )
629:        {
630:           PApp->mDragging = ROI MOVE;
631:        }
632:        else if (xPos >= (ROIRect.right - 4) && xPos <= (ROIRect.right + 4)
633:         && yPos >= (ROIRect.bottom - 4) && yPos <= (ROIRect.bottom + 4) )
634:        {
635:           PApp->mDragging = ROI SIZE;
636:        }
637:        else PApp->mDragging = NULL;
638:        break;
639:     }
640:     // mDragging = ROI MOVE, move the ROI rect to mouse point.
641:     // mDragging = ROI_SIZE, change the size of ROI rect.
642:     case WM_MOUSEMOVE:
643:     {
644:        if (PApp->mDragging == ROI_MOVE || PApp->mDragging == ROI_SIZE)
645:        {
646:           int xPos,yPos;
647:
648:           xPos = LOWORD (lParam);
649:           yPos = HIWORD (lParam);
650:
651:           UpdateRect (hWnd, xPos, yPos);
652:        }
653:        break;
654:     }
655:     case WM_LBUTTONUP:
656:     {
657:        int xPos,yPos;
658:        xPos = LOWORD (lParam);
659:        yPos = HIWORD (lParam);
660:        if (PApp->mDragging == ROI_MOVE)
661:        {
662:           ROIRect.left = xPos;
663:           ROIRect.top  = yPos;
664:        }
665:        else if (PApp->mDragging == ROI_SIZE)
```

```
666:        {
667:           ROIRect.right  = xPos;
668:           ROIRect.bottom = yPos;
669:        }
670:        else;
671:        PApp->mDragging = NULL;
672:        break;
673:     }
674:      case WM_DESTROY:
675:        /*
676:          End display and unregister the video window before it is destroyed.
677:        */
678:        CVID end display(oh Video);
679:     #ifdef VP54
680:        CVP54 unregister video window(oh Resource, oh Video, (int)hWnd);
681:     #else
682:        CVP52 unregister video window(oh Resource, oh Video, (int)hWnd);
683:     #endif
684:        PostQuitMessage(0);
685:        break;
686:
687:      default:
688:        return DefWindowProc(hWnd, msg, wParam, lParam);
689:    }
690:
691:    return 0;
692: }
693:
694:
695: void UpdateRect (HWND hWnd, int xPos, int yPos)
696: {
697:    HPEN oldPen;
698:    HPEN pen;
699:    HDC dc;
700:    CVt52App *PApp = (CVt52App *) AfxGetApp ();
701:
702:    dc = GetDC (hWnd);
703:    SetROP2 (dc, R2 NOT);
704:    //RECT_draw (dc, &ROIRect);
705:
706:    if (PApp->mDragging == ROI_MOVE)
707:    {
708:       ROIRect.right  += xPos - ROIRect.left;
709:       ROIRect.bottom += yPos - ROIRect.top;
710:       ROIRect.left = xPos;
711:       ROIRect.top  = yPos;
712:    }
713:    else
714:    {
715:       ROIRect.right  = xPos;
716:       ROIRect.bottom = yPos;
717:    }
718:
719:    pen = CreatePen(PS SOLID, 1, RGB(255,0,0));
720:    oldPen = (HPEN) SelectObject(dc, pen);
721:    SetROP2 (dc, R2 COPYPEN);
722:    //RECT_draw (dc, &ROIRect);
723:
724:    SelectObject(dc, oldPen);
725:
726:    ReleaseDC(hWnd, dc);
727:    DeleteObject(pen);
728:
729: }
730:
731: void RECT_draw (HDC dc, RECT *ROI)
732: {
733:    MoveToEx(dc, ROI->left,   ROI->top, NULL);
734:    LineTo  (dc, ROI->left,   ROI->bottom);
735:    MoveToEx(dc, ROI->left+1, ROI->top, NULL);
736:    LineTo  (dc, ROI->left+1, ROI->bottom);
737:
738:    MoveToEx(dc, ROI->left,  ROI->top,   NULL);
739:    LineTo  (dc, ROI->right, ROI->top);
740:    MoveToEx(dc, ROI->left,  ROI->top+1, NULL);
```

```
   741:     LineTo  (dc, ROI->right, ROI->top+1);
   742:
   743:     MoveToEx(dc, ROI->right,   ROI->top, NULL);
   744:     LineTo  (dc, ROI->right,   ROI->bottom);
   745:     MoveToEx(dc, ROI->right+1, ROI->top, NULL);
   746:     LineTo  (dc, ROI->right+1, ROI->bottom);
   747:
   748:     MoveToEx(dc, ROI->left,  ROI->bottom,   NULL);
   749:     LineTo  (dc, ROI->right, ROI->bottom);
   750:     MoveToEx(dc, ROI->left,  ROI->bottom+1, NULL);
   751:     LineTo  (dc, ROI->right, ROI->bottom+1);
   752: }
   753:
   754:
   755: void CVt52App::OnImageRead()
   756: {
   757:     // TODO: Add your command handler code here
   758:     CFileDialog OpenDlg (TRUE, "bmp", "*.bmp");
   759:
   760:     if ( OpenDlg.DoModal () != IDOK )
   761:        return;
   762:
   763:     // Set region of interest to acquire.  This is only needed if you
   764:     // desire to override the values contained in the setup file.
   765:
   766:     if (DISPLAYCONSOLE == 1)
   767:        CPB free ( windowHandle [0] );
   768:     windowHandle [0] = CDIB to pb file ( OpenDlg.GetPathName () );
   769:     {
   770:        // Process the acquired window.
   771:        if (DISPLAYCONSOLE == 0)
   772:        {
   773:           oh Console [0] =
   774:                    CIC alloc("Test Console", 300, 300, 300, 300,
   775:                       CDE STYLE CREATE NORMAL);
   776:           DISPLAYCONSOLE = 1;
   777:        }
   778:        CIC clear overlays ( oh Console[0] );
   779:        CIC show image(   oh Console [0], windowHandle [0], 0, 0 );
   780:     }
   781:
   782:     snapdone = TRUE;
   783:
   784:     AllMenuEnable ();
   785: }
   786:
   787: void CVt52App::OnImageWrite()
   788: {
   789:     // TODO: Add your command handler code here
   790:     CFileDialog OpenDlg ( FALSE, "bmp", "*.bmp" );
   791:
   792:     if ( OpenDlg.DoModal () == IDOK )
   793:        CDIB from pb file ( windowHandle [0], OpenDlg.GetPathName () );
   794: }
   795:
   796: void CVt52App::OnImagegrab()
   797: {
   798:     if (SNAP () == 1 )
   799:        AllMenuEnable ();
   800: }
   801:
   802: int CVt52App::SNAP ()
   803: {
   804:     // TODO: Add your command handler code here
   805:     c Int32 acquireTag;
   806:
   807:     // Set region of interest to acquire.  This is only needed if you
   808:     // desire to override the values contained in the setup file.
   809:
   810:     if (DISPLAYCONSOLE == 1)
   811:        CPB free ( windowHandle [0] );
   812:      // Check if an acquire can be started.
   813:     if (!CACQ_channel_can_start (oh_Channel))
   814:     {
   815:        CTC write("Error:  Resources not available for acquire.\n",
```

```
816:                CTC COG DEBUG);
817:        return -1;
818:     }
819:
820:     // Start acquire.
821:     acquireTag = CACQ channel start acquire(oh Channel);
822:
823:      // Complete the acquire, get a handle to acquired window.
824:     windowHandle [0] = CACQ channel complete acquire (oh Channel, acquireTag);
825:     {
826:        if (DISPLAYCONSOLE == 0)
827:        {
828:           oh Console [0] = CIC alloc("Test Console", 300, 300, 300,
      300,CDE STYLE CREATE NORMAL);
829:           DISPLAYCONSOLE = 1;
830:        }
831:        CIC clear overlays ( oh Console[0] );
832:        CIC show image(    oh Console [0],
833:                   windowHandle [0], 0, 0 );
834:     }
835:
836:     snapdone = TRUE;
837:
838:     return 1;
839:
840: }
841:
842: void CVt52App::AllMenuEnable ()
843: {
844:     CWnd *pWnd = AfxGetMainWnd ();
845:     CMenu *pMenu = pWnd->GetMenu ();
846:
847:     pMenu->EnableMenuItem ( MENUFILE,     MF BYPOSITION | MF ENABLED );
848:     pMenu->EnableMenuItem ( MENUCAMERA,   MF BYPOSITION | MF ENABLED );
849:     pMenu->EnableMenuItem ( MENUCALIB,    MF BYPOSITION | MF ENABLED );
850:     pMenu->EnableMenuItem ( MENUSNAP,     MF BYPOSITION | MF ENABLED );
851:     pMenu->EnableMenuItem ( MENUTOOL,     MF BYPOSITION | MF ENABLED );
852:     pMenu->EnableMenuItem ( MENUINSPECT, MF BYPOSITION | MF ENABLED );
853:     ::DrawMenuBar ( pWnd->m hWnd );
854: }
855:
856: void CVt52App::OnUpdateImagegrab(CCmdUI* pCmdUI)
857: {
858:     // TODO: Add your command update UI handler code here
859:     pCmdUI->Enable ( SelectDone && !AutoDone );
860: }
861:
862: void CVt52App::OnUpdateImageWrite(CCmdUI* pCmdUI)
863: {
864:     // TODO: Add your command update UI handler code here
865:      pCmdUI->Enable ( SelectDone && !AutoDone );
866: }
867:
868: void CVt52App::OnRs232c ()
869: {
870:     RS232Done = TRUE;
871:     mCmdBuf.Empty ();
872:     MenuControl ( MENUCOMM, DISABLE );
873:     if ( SelectDone )
874:        MenuControl ( MENURUN,  ENABLE  );
875:     else AUTORUN = 1;
876:
877:     // TODO: Add your command handler code here
878:     AfxBeginThread ( (AFX THREADPROC) CmdInterpret , (LPVOID) this );
879: }
880:
881: c UInt32 CmdInterpret ( LPVOID lptr )
882: {
883:     char TxBuf[300], inchar;
884:     int i;
885:     CVt52App *PApp = (CVt52App *) lptr;
886:     CSingleLock CLock ( &(PApp->mrs232c.mCmdThread) );
887:
888:     AfxMessageBox ( "Start CmdInterpreter", MB OKCANCEL );
889:     for (;;)
```

```
890:     {
891:        CLock.Lock ();
892:        CLock.Unlock ();
893:        PApp->mCmdBuf += (LPSTR) (PApp->mrs232c.RxBuf);
894:        for ( i = 0 ;; i++ )
895:        {
896:           inchar = PApp->mrs232c.RxBuf[i];
897:           if (inchar == 0) break;
898:           else if (inchar == ASCII_CR)
899:           {
900:              if ( PApp->mCmdBuf.Find ( "INSPECT" ) != -1 )
901:              {
902:                 // clear Command
903:                 PApp->mCmdBuf.Empty ();
904:                 // Start Inspection
905:                 if (PApp->mRunLength.RunInspect () != -1 )
906:                 {
907:                    sprintf ( TxBuf, "INSPECT=OK\r" );
908:                    PApp->SendResponse ( TxBuf, 10 );
909:                 }
910:                 else
911:                 {
912:                    sprintf ( TxBuf, "INSPECT=ERROR\r" );
913:                    PApp->SendResponse ( TxBuf, 13 );
914:                 }
915:              }
916:              else if ( PApp->mCmdBuf.Find ( "CHKSTATUS" ) != -1 )
917:              {
918:                 // clear Command
919:                 PApp->mCmdBuf.Empty ();
920:                 // send status
921:                 if ( PApp->VStatus == OFFLINE )
922:                    sprintf ( TxBuf, "CHKSTATUS=1\r");
923:                 else
924:                    sprintf ( TxBuf, "CHKSTATUS=2\r");
925:
926:                 PApp->SendResponse ( TxBuf, 11 );
927:              }
928:              else if ( PApp->mCmdBuf.Find ( "CUTPOINTS" ) != -1 )
929:              {
930:                 // clear Command
931:                 PApp->mCmdBuf.Empty ();
932:                 // send the cutting point
933:                 PApp->ReportCutPoints ();
934:              }
935:              else
936:              {
937:                 // clear Command
938:                 PApp->mCmdBuf.Empty ();
939:                 // send illegal command
940:                 sprintf ( TxBuf, "ILLEGAL COMMAND\r" );
941:                 PApp->SendResponse ( TxBuf, 15 );
942:              }
943:           }
944:        }
945:     }
946:
947:     return 1;
948: }
949:
950: void CVt52App::SendResponse ( char *TxBuf, long count )
951: {
952:     mrs232c.WriteCommBlock( (LPSTR) TxBuf, (DWORD) count )
953: }
954:
955: void CVt52App::OnStart()
956: {
957:     CWnd *pWnd = AfxGetMainWnd ();
958:     CMenu *pMenu = pWnd->GetMenu ();
959:
960:     // TODO: Add your command handler code here
961:     CVID end display ( oh Video );
962:     VStatus = ONLINE;
963:     AutoDone = TRUE;
964:
```

```
   965:     pMenu->EnableMenuItem ( MENUFILE,     MF BYPOSITION | MF DISABLED | MF GRAYE
       D );
   966:     pMenu->EnableMenuItem ( MENUCAMERA,   MF BYPOSITION | MF DISABLED | MF GRAYE
       D );
   967:     pMenu->EnableMenuItem ( MENUCALIB,    MF BYPOSITION | MF DISABLED | MF GRAYE
       D );
   968:     pMenu->EnableMenuItem ( MENUSNAP,     MF BYPOSITION | MF DISABLED | MF GRAYE
       D );
   969:     pMenu->EnableMenuItem ( MENUTOOL,     MF BYPOSITION | MF DISABLED | MF GRAYE
       D );
   970:     pMenu->EnableMenuItem ( MENUINSPECT, MF BYPOSITION | MF DISABLED | MF GRAYE
       D );
   971:
   972:     ::DrawMenuBar ( pWnd->m hWnd );
   973: }
   974:
   975: void CVt52App::OnUpdateStart(CCmdUI* pCmdUI)
   976: {
   977:     pCmdUI->Enable ( !AutoDone && RS232Done );
   978: }
   979:
   980: void CVt52App::OnStop()
   981: {
   982:     // TODO: Add your command handler code here
   983:     CWnd *pWnd = AfxGetMainWnd ();
   984:     CMenu *pMenu = pWnd->GetMenu ();
   985:     CVID begin display(oh Video, &vgaWindow);
   986:
   987:     VStatus = OFFLINE;
   988:     AutoDone = FALSE;
   989:
   990:     pMenu->EnableMenuItem ( MENUFILE,     MF BYPOSITION | MF ENABLED );
   991:     pMenu->EnableMenuItem ( MENUCAMERA,   MF BYPOSITION | MF ENABLED );
   992:     pMenu->EnableMenuItem ( MENUCALIB,    MF BYPOSITION | MF ENABLED );
   993:     pMenu->EnableMenuItem ( MENUSNAP,     MF BYPOSITION | MF ENABLED );
   994:
   995:     ::DrawMenuBar ( pWnd->m hWnd );
   996: }
   997:
   998: void CVt52App::OnUpdateStop(CCmdUI* pCmdUI)
   999: {
  1000:     // TODO: Add your command update UI handler code here
  1001:     pCmdUI->Enable ( AutoDone );
  1002: }
  1003:
  1004: void CVt52App::OnUpdateRs232c(CCmdUI* pCmdUI)
  1005: {
  1006:     pCmdUI->Enable ( !RS232Done );
  1007: }
  1008:
  1009: void CVt52App::OnAuto()
  1010: {
  1011:     // TODO: Add your command handler code here
  1012:     mRunLength.RunInspect ();
  1013: }
  1014:
  1015: void CVt52App::OnUpdateAuto(CCmdUI* pCmdUI)
  1016: {
  1017:     // TODO: Add your command update UI handler code here
  1018:     pCmdUI->Enable ( SelectDone );
  1019: }
  1020:
  1021: void CVt52App::ReportCutPoints ()
  1022: {
  1023:     char TxBuf [512], strbuf[50];
  1024:     int    txpnt;
  1025:     int    checksum;
  1026:     int  i;
  1027:
  1028:     checksum = 0;
  1029:     // clear TxBuf
  1030:     memset ( TxBuf, 0, 512 );
  1031:
  1032:     // add Total Cutting Numbers to TxBuf
  1033:     sprintf ( TxBuf, "C%d ", mcutcnt );
```

```
1034:     checksum += mcutcnt;
1035:
1036:     for ( i = 0 ; i < mcutcnt ; i++ )
1037:     {
1038:        sprintf ( strbuf,"%d %d ", xCutPnt [i], yCutPnt[i] );
1039:        strcat ( TxBuf, strbuf );
1040:        checksum += xCutPnt[i] + yCutPnt[i];
1041:     }
1042:
1043:     txpnt = strlen ( TxBuf );
1044: }
1045:
1046: void CVt52App::OnRobot()
1047: {
1048:     Robot robot;
1049:
1050:     robot.DoModal();
1051: }
1052:
1053: #define      ZPULSEOFFSET       170.0
1054: #define      ZPULSEOFST1        150.0
1055: #define      ZINSTRAYOFST       100.0
1056: #define      ZOUTTRAYOFST       80.0
1057: void CVt52App::OnSupplyFulltray()
1058: {
1059:     // TODO: Add your command handler code here
1060:     double dis;
1061:     int    ret;
1062:
1063:     if ( MotMoveWait ( M_TRZ, -0.5 )  == -1 ) return;
1064:
1065:     FullTrayFixOnOff ( ON ); Sleep ( 300 );
1066:     FullTrayForRev  ( ON ); CoverTrayForRev ( ON );
1067:     Sleep ( 1000 );
1068:     // Check the full tray position
1069:     // Open the Full Tray's cover
1070:     if ( !ReadSensor ( TRZHOMEPOS ) )
1071:     {
1072:        ret = ZNotInHomeMsgBox ();
1073:        if ( ret == IDCANCEL ) return;
1074:     }
1075:
1076:     if ( MotMoveWait ( M_TRX, CSysData::m_dtrx_pos1 ) == -1 ) return;
1077:     GripperOff ();
1078:     CoverClose (); Sleep ( 500 );
1079:
1080:     if ( MotMoveWait ( M_TRZ, CSysData::m_dtrz_pos1 ) == -1 ) return;
1081:     Sleep (400) ; GripperOn ();
1082:     Sleep ( 400 ); CoverOpen (); Sleep ( 1000 );
1083:     if ( MotMoveWait ( M_TRZ, -0.5 )  == -1 ) return;
1084:     Sleep (400);
1085:     // move the opened cover to COVER Stage
1086:     if ( !ReadSensor ( TRZHOMEPOS ) )
1087:     {
1088:        ret = ZNotInHomeMsgBox ();
1089:        if ( ret == IDCANCEL ) return;
1090:     }
1091:     if ( MotMoveWait ( M_TRX, CSysData::m_dtrx_pos2 ) == -1 ) return;
1092:     if ( MotMoveWait ( M_TRZ, CSysData::m_dtrz_pos2 ) == -1 ) return;
1093:     Sleep ( 1000 );   GripperOff (); Sleep ( 500 );
1094:     if ( MotMoveWait ( M_TRZ, 0.0 ) == -1 ) return;
1095:
1096:     // move the full tray position
1097:     CoverClose (); Sleep ( 500 );
1098:     if ( !ReadSensor ( TRZHOMEPOS ) )
1099:     {
1100:        ret = ZNotInHomeMsgBox ();
1101:        if ( ret == IDCANCEL ) return;
1102:     }
1103:     if ( MotMoveWait ( M_TRX, CSysData::m_dtrx_pos1 ) == -1 ) return;
1104:     dis = ZPULSEOFFSET * CPpc2310App::m_ndEncUnit[M_TRZ];
1105:     if ( MotMoveWait ( M_TRZ, CSysData::m_dtrz_pos1 + dis ) == -1 ) return;
1106:     Sleep ( 2000 ); GripperOn (); Sleep ( 500 );
1107:     FullTrayFixOnOff ( OFF );
1108:     // move the full tray to inspection stage
```

```
1109:     InspectTrayFixOnOff ( OFF ); Sleep ( 600 );
1110:     if ( MotMoveWait ( M_TRZ, -0.5 ) == -1 ) return;
1111:     Sleep (400);
1112:     if ( !ReadSensor ( TRZHOMEPOS ) )
1113:     {
1114:        ret = ZNotInHomeMsgBox ();
1115:        if ( ret == IDCANCEL ) return;
1116:     }
1117:     if ( MotMoveWait ( M_TRX, CSysData::m_dtrx_pos3 ) == -1 ) return;
1118:     if ( MotMoveWait ( M_TRZ, CSysData::m_dtrz_pos3 ) == -1 ) return;
1119:     Sleep ( 2000 ); GripperOff (); Sleep ( 500 );
1120:     InspectTrayFixOnOff ( ON ); CoverTrayForRev ( OFF );
1121:
1122:     // move to COVER Stage
1123:     if ( MotMoveWait ( M_TRZ, -0.5 ) == -1 ) return;
1124:     Sleep (400);
1125:     if ( !ReadSensor ( TRZHOMEPOS ) )
1126:     {
1127:        ret = ZNotInHomeMsgBox ();
1128:        if ( ret == IDCANCEL ) return;
1129:     }
1130:     if ( MotMoveWait ( M_TRX, CSysData::m_dtrx_pos5 ) == -1 ) return;
1131: }
1132:
1133: void CVt52App::OnSupplyEmptytray()
1134: {
1135:     int ret;
1136:
1137:     // TODO: Add your command handler code here
1138:     if ( MotMoveWait ( M_TRZ, -0.5 ) == -1 ) return;
1139:     EmptyTrayFixOnOff ( ON ); Sleep ( 500 );
1140:     EmptyTrayForRev   ( ON ); Sleep ( 300 );
1141:     // Check the full tray position
1142:     // Open the Full Tray's cover
1143:     if ( !ReadSensor ( TRZHOMEPOS ) )
1144:     {
1145:        ret = ZNotInHomeMsgBox ();
1146:        if ( ret == IDCANCEL ) return;
1147:     }
1148:     if ( MotMoveWait ( M_TRX, CSysData::m_dtrx_pos4 ) == -1 ) return;
1149:     GripperOff (); CoverClose (); Sleep ( 300 );
1150:     if ( MotMoveWait ( M_TRZ, CSysData::m_dtrz_pos4 ) == -1 ) return;
1151:     GripperOn (); Sleep ( 300 ); CoverOpen ();
1152:
1153:     // move the opened cover to COVER Stage
1154:     OutTrayForRev ( ON ); Sleep ( 1000 );
1155:     if ( MotMoveWait ( M_TRZ, -0.5 ) == -1 ) return;
1156:     Sleep (400);
1157:     if ( !ReadSensor ( TRZHOMEPOS ) )
1158:     {
1159:        ret = ZNotInHomeMsgBox ();
1160:        if ( ret == IDCANCEL ) return;
1161:     }
1162:     if ( MotMoveWait ( M_TRX, CSysData::m_dtrx_pos5 ) == -1 ) return;
1163:     if ( MotMoveWait ( M_TRZ, CSysData::m_dtrz_pos5 ) == -1 ) return;
1164:     Sleep ( 1000 );   GripperOff (); Sleep ( 1000 );
1165:     if ( MotMoveWait ( M_TRZ, -0.5 ) == -1 ) return;
1166:
1167:     // move the out tray position
1168:     Sleep ( 500 );
1169:     if ( !ReadSensor ( TRZHOMEPOS ) )
1170:     {
1171:        ret = ZNotInHomeMsgBox ();
1172:        if ( ret == IDCANCEL ) return;
1173:     }
1174:     CoverClose ();
1175:     Sleep (500);
1176:     if ( MotMoveWait ( M_TRX, CSysData::m_dtrx_pos5 ) == -1 ) return;
1177: }
1178:
1179: void CVt52App::OnOutputCuttray()
1180: {
1181:     double dis;
1182:     int    ret;
1183:
```

```
1184:     // TODO: Add your command handler code here
1185:     if ( MotMoveWait ( M_TRZ, -0.5 ) == -1 ) return;
1186:     OutTrayForRev ( ON ); EmptyTrayFixOnOff ( ON ); Sleep (500);
1187:
1188:     // Check the Out tray position
1189:     // pickup the Empty Tray's cover
1190:     GripperOff (); CoverOpen (); Sleep ( 300 );
1191:     if ( !ReadSensor ( TRZHOMEPOS ) )
1192:     {
1193:        ret = ZNotInHomeMsgBox ();
1194:        if ( ret == IDCANCEL ) return;
1195:     }
1196:     if ( MotMoveWait ( M_TRX, CSysData::m_dtrx_pos5 ) == -1 ) return;
1197:     if ( MotMoveWait ( M_TRZ, CSysData::m_dtrz_pos5 ) == -1 ) return;
1198:     GripperOn (); Sleep ( 500 );
1199:     if ( MotMoveWait ( M_TRZ, -0.5 )  == -1 ) return;
1200:     // move the opened cover to COVER Stage
1201:     Sleep (400);
1202:     if ( !ReadSensor ( TRZHOMEPOS ) )
1203:     {
1204:        ret = ZNotInHomeMsgBox ();
1205:        if ( ret == IDCANCEL ) return;
1206:     }
1207:     if ( MotMoveWait ( M_TRX, CSysData::m_dtrx_pos4 ) == -1 ) return;
1208:     if ( MotMoveWait ( M_TRZ, CSysData::m_dtrz_pos4 ) == -1 ) return;
1209:     Sleep ( 1000 );   CoverClose (); Sleep ( 500 ); GripperOff (); Sleep ( 500
      );
1210:     EmptyTrayFixOnOff ( OFF ); Sleep ( 1000 );
1211:     dis = ZPULSEOFST1 * CPpc2310App::m_ndEncUnit[M_TRZ];
1212:     if ( MotMoveWait ( M_TRZ, CSysData::m_dtrz_pos4 + dis) == -1 ) return;
1213:     Sleep (500);  GripperOn (); Sleep (600);
1214:     if ( MotMoveWait ( M_TRZ, -0.5 ) == -1 ) return;
1215:
1216:     // move the full tray position
1217:     Sleep (400);
1218:     if ( !ReadSensor ( TRZHOMEPOS ) )
1219:     {
1220:        ret = ZNotInHomeMsgBox ();
1221:        if ( ret == IDCANCEL ) return;
1222:     }
1223:     if ( MotMoveWait ( M_TRX, CSysData::m_dtrx_pos5 ) == -1 ) return;
1224:     dis = ZOUTTRAYOFST * CPpc2310App::m_ndEncUnit[M_TRZ];
1225:     dis = CSysData::m_dtrz_pos5 - dis;
1226:     if ( MotMoveWait ( M_TRZ, dis ) == -1 ) return;
1227:     Sleep ( 1000 );
1228:     GripperOff (); Sleep ( 1000 );
1229:
1230:     // move the full tray to inspectuion stage
1231:     if ( MotMoveWait ( M_TRZ, -0.5 ) == -1 ) return;
1232:     OutTrayForRev ( OFF );
1233:
1234: }
1235:
1236: void CVt52App::OnOutputInstray()
1237: {
1238:     double dis;
1239:     int    ret;
1240:
1241:     // TODO: Add your command handler code here
1242:     if ( MotMoveWait ( M_TRZ, -0.5 ) == -1 ) return;
1243:     Sleep (500);
1244:     GripperOff (); CoverClose (); Sleep ( 300 );
1245:     InspectTrayFixOnOff ( OFF ); CoverTrayForRev ( ON ); Sleep ( 500 );
1246:
1247:     // Check the Out tray position
1248:     // pickup the Empty Tray's cover
1249:     if ( !ReadSensor ( TRZHOMEPOS ) )
1250:     {
1251:        ret = ZNotInHomeMsgBox ();
1252:        if ( ret == IDCANCEL ) return;
1253:     }
1254:     if ( MotMoveWait ( M_TRX, CSysData::m_dtrx_pos3 ) == -1 ) return;
1255:     if ( MotMoveWait ( M_TRZ, CSysData::m_dtrz_pos3 ) == -1 ) return;
1256:     GripperOn (); Sleep ( 500 );
1257:     if ( MotMoveWait ( M_TRZ, -0.5 )  == -1 ) return;
```

```
1258:
1259:     // move the opened cover to COVER Stage
1260:     Sleep (400);
1261:     if ( !ReadSensor ( TRZHOMEPOS ) )
1262:     {
1263:        ret = ZNotInHomeMsgBox ();
1264:        if ( ret == IDCANCEL ) return;
1265:     }
1266:     if ( MotMoveWait ( M_TRX, CSysData::m_dtrx_pos2 ) == -1 ) return;
1267:     dis = ZINSTRAYOFST * CPpc2310App::m_ndEncUnit[M_TRZ];
1268:     if ( MotMoveWait ( M_TRZ, CSysData::m_dtrz_pos2 - dis ) == -1 ) return;
1269:     Sleep ( 1000 );   GripperOff (); Sleep ( 1000 );
1270:     if ( MotMoveWait ( M_TRZ, -0.5 ) == -1 ) return;
1271:
1272:     // move the full tray position
1273:     Sleep (400);
1274:     if ( !ReadSensor ( TRZHOMEPOS ) )
1275:     {
1276:        ret = ZNotInHomeMsgBox ();
1277:        if ( ret == IDCANCEL ) return;
1278:     }
1279:     if ( MotMoveWait ( M_TRX, CSysData::m_dtrx_pos2 ) == -1 ) return;
1280:     CoverTrayForRev ( OFF ); Sleep ( 1000 );
1281: }
1282:
1283: int CVt52App::ZNotInHomeMsgBox ()
1284: {
1285:     int ret;
1286:
1287:     ret = AfxMessageBox ( " Z축이 Home 위치에서 벗어 났읍니다 ", MB_OKCANCEL );
1288:
1289:     return ret;
1290: }
1291:
1292: BOOL CVt52App::ReadSensor ( int sensor )
1293: {
1294:     BYTE axis_status1, axis_status2, axis_status3;
1295:     BYTE ONOFF;
1296:
1297:     CPpc2310App::Monitor ();
1298:     // X 및 Z 축 Sensor
1299:     axis_status1 = CPpc2310App::m_nInputSignalDummy [M_TRX];
1300:     axis_status2 = CPpc2310App::m_nInputSignalDummy [M_TRZ];
1301:     axis_status3 = CPpc2310App::m_nInputSignalDummy [M_DUMMY];
1302:
1303:     switch ( sensor )
1304:     {
1305:        case TRXMLIMPOS :
1306:           if ( axis_status1 & 8 )   ONOFF = ON;
1307:           else                      ONOFF = OFF;
1308:           break;
1309:        case TRXHOMEPOS :
1310:           if ( axis_status1 & 2 ) ONOFF = ON;
1311:           else                    ONOFF = OFF;
1312:           break;
1313:        case TRXPLIMPOS :
1314:           if ( axis_status1 & 0x10 ) ONOFF = ON;
1315:           else                       ONOFF = OFF;
1316:           break;
1317:        case TRZHOMEPOS :
1318:           if ( axis_status2 & 2 ) ONOFF = ON;
1319:           else                    ONOFF = OFF;
1320:           break;
1321:        case CHUCKPOS :
1322:           if ( axis_status1 & 0x40 ) ONOFF = ON;
1323:           else                       ONOFF = OFF;
1324:           break;
1325:        case FULLPOS  :
1326:           if ( axis_status2 & 4 ) ONOFF = ON;
1327:           else                    ONOFF = OFF;
1328:           break;
1329:        case COVERPOS :
1330:           if ( axis_status3 & 0x40)  ONOFF = ON;
1331:           else                       ONOFF = OFF;
1332:           break;
```

```
1333:        case EMPTYPOS    :
1334:           if ( axis status1 & 4 )     ONOFF = ON;
1335:           else                        ONOFF = OFF;
1336:           break;
1337:        case OUTPOS    :
1338:           if ( axis status2 & 0x40 ) ONOFF = ON;
1339:           else                        ONOFF = OFF;
1340:           break;
1341:        case ROBOTIN:  // Robot
1342:           if ( axis status3 & 4 ) ONOFF = ON;
1343:           else                ONOFF = OFF;
1344:           break;
1345:     }
1346:
1347:     return (ONOFF);
1348: }
1349:
1350: void CVt52App::OnUpdateOutputCuttray(CCmdUI* pCmdUI)
1351: {
1352:     // TODO: Add your command update UI handler code here
1353:     pCmdUI->Enable ( HOMEDONE );
1354: }
1355:
1356: void CVt52App::OnUpdateOutputInstray(CCmdUI* pCmdUI)
1357: {
1358:     // TODO: Add your command update UI handler code here
1359:     pCmdUI->Enable ( HOMEDONE );
1360: }
1361:
1362: void CVt52App::OnUpdateSupplyEmptytray(CCmdUI* pCmdUI)
1363: {
1364:     // TODO: Add your command update UI handler code here
1365:     pCmdUI->Enable ( HOMEDONE );
1366: }
1367:
1368: void CVt52App::OnUpdateSupplyFulltray(CCmdUI* pCmdUI)
1369: {
1370:     // TODO: Add your command update UI handler code here
1371:     pCmdUI->Enable ( HOMEDONE );
1372: }
1373:
1374: void CVt52App::OnUpdateTrayposSetup(CCmdUI* pCmdUI)
1375: {
1376:     // TODO: Add your command update UI handler code here
1377:     pCmdUI->Enable ( HOMEDONE );
1378: }
```

```
    1: // Robot.cpp : implementation file
    2: //
    3:
    4: #include "stdafx.h"
    5: #include "vt52.h"
    6: #include "global.h"
    7: #include "Robot.h"
    8: #include "Command.h"
    9: #include "PortSet.h"
   10:
   11: UINT  cur Category, cur Command, cur Line No = 0;
   12:
   13: CString  m strSendText;     // Messages to send
   14: CString  m strText;         // Message to Compose
   15: CString  m Vision;          // Messages from Vision System
   16: CString m ReceiveData;      // Message from Robot Controller
   17:
   18: int       length;
   19: int       ReceiveFlag = FALSE;
   20: char  mes[100];
   21: BOOL  vision flag=FALSE;   // Check whether vision included
   22: FILE  *in;
   23: BYTE  EXPort = 1;        // Set COM Port to COM 2
   24: extern    COMMCONFIG  cc;
   25:
   26: /////////////////////////////////////////////////////////////////////////////
   27: // Robot dialog
   28:
   29:
   30: Robot::Robot(CWnd* pParent /*=NULL*/)
   31:     : CDialog(Robot::IDD, pParent)
   32: {
   33:     //{{AFX_DATA_INIT(Robot)
   34:     m Comment = _T("");
   35:     m Program = _T("");
   36:     m Program No = 0;
   37:     //}}AFX_DATA_INIT
   38: }
   39:
   40:
   41: void Robot::DoDataExchange(CDataExchange* pDX)
   42: {
   43:     CDialog::DoDataExchange(pDX);
   44:     //{{AFX_DATA_MAP(Robot)
   45:     DDX_Control(pDX, IDC_COMMAND_COMBO, m_CommandCombo);
   46:     DDX_Control(pDX, IDC_CATEGORY_COMBO, m_CategoryCombo);
   47:     DDX_Control(pDX, IDC_ARGUMENT_COMBO, m_ArgumentCombo);
   48:     DDX_Text(pDX, IDC_COMMENT_EDIT, m_Comment);
   49:     DDX_Text(pDX, IDC_PROGRAM_EDIT, m_Program);
   50:     DDX_Text(pDX, IDC_NO_EDIT, m_Program_No);
   51:     //}}AFX_DATA_MAP
   52: }
   53:
   54:
   55: BEGIN_MESSAGE_MAP(Robot, CDialog)
   56:     //{{AFX_MSG_MAP(Robot)
   57:     ON_CBN_SELCHANGE(IDC_CATEGORY_COMBO, OnSelchangeCategoryCombo)
   58:     ON_CBN_SELCHANGE(IDC_COMMAND_COMBO, OnSelchangeCommandCombo)
   59:     ON_CBN_EDITCHANGE(IDC_ARGUMENT_COMBO, OnEditchangeArgumentCombo)
   60:     ON_CBN_SELCHANGE(IDC_ARGUMENT_COMBO, OnSelchangeArgumentCombo)
   61:     ON_BN_CLICKED(IDC_COMPLETE_BUTTON, OnCompleteButton)
   62:     ON_EN_CHANGE(IDC_NO_EDIT, OnChangeNoEdit)
   63:     ON_EN_CHANGE(IDC_PROGRAM_EDIT, OnChangeProgramEdit)
   64:     ON_BN_CLICKED(IDC_SEND_BUTTON, OnSendButton)
   65:     ON_BN_CLICKED(IDC_START_BUTTON, OnStartButton)
   66:     //}}AFX_MSG_MAP
   67:     ON_MESSAGE(WM_RECEIVEDATA, OnReceiveData)
   68: END_MESSAGE_MAP()
   69:
   70: /////////////////////////////////////////////////////////////////////////////
   71: // Robot message handlers
   72:
   73: BOOL Robot::OnInitDialog()
   74: {
   75:     CDialog::OnInitDialog();
```

```
   76:
   77:     char  arg[100];
   78:
   79:     m Comm.SetComPort(1, 9600, 8,2,2);
   80:     m Comm.CreateCommInfo();
   81:     m Comm.OpenComPort();
   82:     m Comm.SetHwnd(this->m_hWnd);
   83:
   84:     in = fopen("args.ini", "a+");     // for the argument list history
   85:
   86:     for (;;) {
   87:        if (fgets(arg, 100, in) == NULL)
   88:           break;
   89:        arg[strlen(arg)-1] = NULL;
   90:        m ArgumentCombo.AddString(arg);
   91:     }
   92:
   93:     for (int i=0 ; i<36 ; i++)
   94:        m CommandCombo.AddString(m Command[0][i]);
   95:
   96:     memset(mes, 0, 100);
   97:
   98:     cur Category = 0;
   99:     cur Line No += 10;
  100:     cur Command = 12;        // Move Position Command
  101:     m Program No = cur Line No;
  102:     m CategoryCombo.SetCurSel(cur Category);
  103:     m CommandCombo.SetCurSel(cur Command);
  104:     m Comment = m Comment1[cur Category][cur Command];
  105:     UpdateData(FALSE);
  106:
  107:     return TRUE;  // return TRUE unless you set the focus to a control
  108:                   // EXCEPTION: OCX Property Pages should return FALSE
  109: }
  110:
  111: void Robot::OnOK()
  112: {
  113:     fclose(in);
  114:     CDialog::OnOK();
  115: }
  116:
  117: void Robot::OnCancel()
  118: {
  119:     fclose(in);
  120:     CDialog::OnCancel();
  121: }
  122:
  123: void Robot::OnSelchangeCategoryCombo()
  124: {
  125:     cur Category = m CategoryCombo.GetCurSel();
  126:     cur Command = 0;
  127:
  128:     m CommandCombo.ResetContent();       // Clear Entire Contents of Lists
  129:
  130:     for (int i=0 ; i<36 ; i++) {
  131:        if (m_Command[cur_Category][i][0] != 0)       // Some Lists contain null
      char
  132:           m CommandCombo.AddString(m Command[cur Category][i]);
  133:     }
  134:
  135:     m CommandCombo.SetCurSel(cur Command);
  136:     m Comment = m Comment1[cur Category][cur Command];
  137:     UpdateData(FALSE);
  138: }
  139:
  140: void Robot::OnSelchangeCommandCombo()
  141: {
  142:     cur Command = m CommandCombo.GetCurSel();
  143:     m Comment = m Comment1[cur Category][cur Command];
  144:     UpdateData(FALSE);
  145: }
  146:
  147: void Robot::OnEditchangeArgumentCombo()
  148: {
  149:     m ArgumentCombo.GetWindowText(mes, 100);
```

```
150: }
151:
152: void Robot::OnSelchangeArgumentCombo()
153: {
154:    m_ArgumentCombo.GetLBText(m_ArgumentCombo.GetCurSel(), mes);
155: }
156:
157: void Robot::OnCompleteButton()
158: {
159:    if (mes[0]!=0 && m_ArgumentCombo.FindStringExact(-1, mes)==CB_ERR) {
160:       fprintf(in, "%s\n", mes);
161:       m_ArgumentCombo.AddString(mes);
162:    }
163:
164:    m_strText.Format("%d %s %s\r\n", m_Program_No,
     m_Command[cur_Category][cur_Command], mes);
165:    cur_Line_No += 10;
166:    m_Program_No = cur_Line_No;
167:    m_Program += m_strText;
168:    UpdateData(FALSE);
169: }
170:
171: void Robot::OnChangeNoEdit()
172: {
173:    UpdateData(TRUE);
174: }
175:
176: void Robot::OnChangeProgramEdit()
177: {
178:    UpdateData(TRUE);
179: }
180:
181: void Robot::OnSendButton()
182: {
183:    char  *tmp;
184:    int      len, i;
185:
186:       m_strSendText = m_Program;
187:       len = m_strSendText.GetLength();
188:       tmp = (char *)malloc(len+1);
189:
190:       int j=0;
191:       for (i=0 ; i<len ; i++) {
192:          if ( m_strSendText[i] != 0x0a )
193:             tmp[j++] = m_strSendText[i];
194:          if ( m_strSendText[i] == 0x0d ) {
195:             m_Comm.WriteCommBlock(tmp, j);
196:             Sleep(700);
197:             j = 0;
198:          }
199:       }
200:       free(tmp);
201: }
202:
203: LONG Robot::OnReceiveData(UINT WParam, LONG LParam)
204: {
205:    m_ReceiveData += (LPSTR)m_Comm.RxBuf;
206:    ReceiveFlag = TRUE;
207:    return TRUE;
208: }
209:
210: void Robot::OnStartButton()
211: {
212:    CVt52App *PApp = (CVt52App *)AfxGetApp();
213:
214:    PApp->OnSupplyFulltray();      // 배양 트레이 공급
215:    FullTrayForRev(0);
216:    Sleep(1000);
217:    PApp->OnSupplyEmptytray();     // Empty Tray 공급
218:
219:    SendVisData();
220:    AfxMessageBox("Vision Processing Ended!", MB_OK);
221:    PApp->OnOutputInstray();       // Inspect Tray 배출
222:    Sleep(1000);
223:    PApp->OnOutputCuttray();       // Cutting Tray 배출
```

```
224:    EmptyTrayForRev(0);
225: }
226:
227: void Robot::SendVisData(void)
228: {
229:    CVt52App *PApp = (CVt52App *) AfxGetApp ();
230:    CString     m_data;
231:    char      *tmp;
232:    int         i, len;
233:    int         orig[2] = { 389, 95 };  // origin position
234:    // Origin : 388.92, 94.68, 333.51, 178.92, 1.22, -90.46
235:
236:    m_data.Empty();
237:
238:    if (PApp->mRunLength.RunInspect () == -1) {
239:       AfxMessageBox("Vision Inspection Error!!!",MB_OK);
240:       return;
241:    }
242:
243:    m_data.Format("PRN %d\r\n",PApp->mcutcnt);    // Send # of total cut points
244:
245:    CString VisionPos;
246:
247:    VisionPos.Empty();
248:
249:    for (i=0 ; i<PApp->mcutcnt ; i++) {
250:       VisionPos.Format("PRN %d, %d, 290, 178.92, 1.22, -90.46\r\n",
     orig[0]+PApp->xCutPnt[i], orig[1]+PApp->yCutPnt[i]);
251:       m_data += VisionPos;
252:    }
253:    AfxMessageBox(m_data,MB_OK);
254:    len = m_data.GetLength();
255:    tmp = (char *)malloc(len+1);
256:
257:    int j=0;
258:    for (i=0 ; i<len ; i++) {
259:       if ( m_data[i] != 0x0a )
260:          tmp[j++] = m_data[i];
261:       if ( m_data[i] == 0x0d ) {
262:          m_Comm.WriteCommBlock(tmp, j);
263:          Sleep(700);
264:          j = 0;
265:       }
266:    }
267:    free(tmp);
268: }
```

```
    1: // RunLength.cpp : implementation file
    2: //
    3:
    4: #include "stdafx.h"
    5: #include "vt52.h"
    6: #include "vision.h"
    7: #include "RunLength.h"
    8:
    9: extern CACQ resource oh oh Resource;
   10: extern CACQ channel oh  oh_Channel;
   11: extern CVID oh oh Video;
   12: extern CPB oh  windowHandle[MAX FRAME];
   13: extern CIC oh  oh Console[MAX_FRAME];
   14: extern RECT ROIRect;
   15:
   16: BOOL edge type ( CPB oh, int, int, int, short);
   17: /////////////////////////////////////////////////////////////////////////////
   18: // CRunLength dialog
   19:
   20:
   21: CRunLength::CRunLength(CWnd* pParent /*=NULL*/)
   22:    : CDialog(CRunLength::IDD, pParent)
   23: {
   24:    //{{AFX DATA INIT(CRunLength)
   25:    // m xorg    = ROIRect.left;
   26:    //m xsize   = ROIRect.right - ROIRect.left;
   27:    //m yorg    = ROIRect.top;
   28:    //m ysize   = ROIRect.bottom - ROIRect.top;
   29:    //m sample = 5;
   30:    //m diff = 20;
   31:    //thres = 90;
   32:    //}}AFX_DATA_INIT
   33: }
   34:
   35:
   36: void CRunLength::DoDataExchange(CDataExchange* pDX)
   37: {
   38:    CDialog::DoDataExchange(pDX);
   39:    //{{AFX DATA MAP(CRunLength)
   40:    DDX Text(pDX, IDC RUN XORG, m xorg);
   41:    DDV MinMaxInt(pDX, m xorg, 0, 640);
   42:    DDX Text(pDX, IDC RUN XSIZE, m xsize);
   43:    DDV MinMaxInt(pDX, m xsize, 0, 640);
   44:    DDX Text(pDX, IDC RUN YORG, m yorg);
   45:    DDV MinMaxInt(pDX, m yorg, 0, 480);
   46:    DDX Text(pDX, IDC RUN YSIZE, m ysize);
   47:    DDV MinMaxInt(pDX, m ysize, 0, 480);
   48:    DDX Text(pDX, IDC SAMPLE FACTOR, m sample);
   49:    DDV MinMaxInt(pDX, m sample, 0, 30);
   50:    DDX Text(pDX, IDC EDGE DIFF, m diff);
   51:    DDV MinMaxInt(pDX, m diff, 0, 255);
   52:    //}}AFX_DATA_MAP
   53: }
   54:
   55:
   56: BEGIN MESSAGE MAP(CRunLength, CDialog)
   57:    //{{AFX MSG MAP(CRunLength)
   58:    ON EN CHANGE(IDC RUN XORG, OnChangeRunXorg)
   59:    ON EN CHANGE(IDC RUN XSIZE, OnChangeRunXsize)
   60:    ON EN CHANGE(IDC RUN YORG, OnChangeRunYorg)
   61:    ON EN CHANGE(IDC RUN YSIZE, OnChangeRunYsize)
   62:    ON EN CHANGE(IDC SAMPLE FACTOR, OnChangeSampleFactor)
   63:    ON BN CLICKED(ID RUNLENGTH, OnRunlength)
   64:    ON EN CHANGE(IDC EDGE DIFF, OnChangeEdgeDiff)
   65:    ON BN CLICKED(ID RUNLENGTH2, OnRunlength2)
   66:    ON BN CLICKED(ID RUNLENGTH3, OnRunlength3)
   67:    ON BN CLICKED(ID RUNLENGTH4, OnRunlength4)
   68:    //}}AFX MSG MAP
   69: END MESSAGE MAP()
   70:
   71: /////////////////////////////////////////////////////////////////////////////
   72: // CRunLength message handlers
   73:
   74: void CRunLength::OnChangeRunXorg()
   75: {
```

```
   76:     int temp;
   77:
   78:     CVt52App *PApp = (CVt52App *) AfxGetApp ();
   79:
   80:     if (vdebug) CIC clear overlays ( oh Console[PApp->Frame_Index] );
   81:     temp = (int) GetDlgItemInt (IDC_RUN_XORG);
   82:
   83:     if ( temp  < 0 || ( temp + m_xsize ) >= 640 )
   84:     {
   85:        AfxMessageBox ( " xorg > 0 이고, \r\n xorg + xsize < 640 이어야 합니다"
      );
   86:        return;
   87:     }
   88:
   89:     ROIRect.left = m_xorg = temp;
   90:     if (vdebug) CIC draw rect ( oh Console[PApp->Frame Index], (double) m_xorg,
   91:                 (double) m_yorg, (double) m_xsize, (double) m_ysize,
   92:                 CDE PEN GREEN );
   93:     if (vdebug) CIC_force_overlays ( oh_Console[PApp->Frame_Index] );
   94: }
   95:
   96: void CRunLength::OnChangeRunXsize()
   97: {
   98:     int temp;
   99:
  100:     CVt52App *PApp = (CVt52App *) AfxGetApp ();
  101:
  102:     if (vdebug) CIC clear overlays ( oh Console[PApp->Frame_Index] );
  103:     temp = (int) GetDlgItemInt (IDC_RUN_XSIZE);
  104:     if ( ( m_xorg + temp ) >= 640 )
  105:     {
  106:        AfxMessageBox ( " xorg + xsize < 640 이어야 합니다" );
  107:        return;
  108:     }
  109:
  110:     m_xsize = temp;
  111:     ROIRect.right = ROIRect.left + m_xsize;
  112:     if (vdebug) CIC draw rect ( oh Console[PApp->Frame Index], (double) m_xorg,
  113:                 (double) m_yorg, (double) m_xsize, (double) m_ysize,
  114:                 CDE PEN GREEN );
  115:     if (vdebug) CIC_force_overlays ( oh_Console[PApp->Frame_Index] );
  116: }
  117:
  118: void CRunLength::OnChangeRunYorg()
  119: {
  120:     int temp;
  121:
  122:     CVt52App *PApp = (CVt52App *) AfxGetApp ();
  123:
  124:     if (vdebug) CIC clear overlays ( oh Console[PApp->Frame_Index] );
  125:     temp = (int) GetDlgItemInt (IDC_RUN_YORG);
  126:     if ( temp  < 0 || ( temp + m_ysize ) >= 480 )
  127:     {
  128:        AfxMessageBox ( " yorg > 0 이고, \r\n yorg + ysize < 480 이어야 합니다"
      );
  129:        return;
  130:     }
  131:
  132:     ROIRect.top = m_yorg = temp;
  133:     if (vdebug) CIC draw rect ( oh Console[PApp->Frame Index], (double) m_xorg,
  134:                 (double) m_yorg, (double) m_xsize, (double) m_ysize,
  135:                 CDE PEN GREEN );
  136:     if (vdebug) CIC_force_overlays ( oh_Console[PApp->Frame_Index] );
  137: }
  138:
  139: void CRunLength::OnChangeRunYsize()
  140: {
  141:     int temp;
  142:     CVt52App *PApp = (CVt52App *) AfxGetApp ();
  143:
  144:     if (vdebug) CIC clear overlays ( oh Console[PApp->Frame_Index] );
  145:     temp = (int) GetDlgItemInt (IDC_RUN_YSIZE);
  146:     if ( ( m_yorg + temp ) >= 480 )
  147:     {
  148:        AfxMessageBox ( " yorg + ysize < 480 이어야 합니다" );
```

```
149:        return;
150:     }
151:
152:     m ysize = temp;
153:     ROIRect.bottom = ROIRect.top + m ysize;
154:     if (vdebug) CIC draw rect ( oh Console[PApp->Frame Index], (double) m_xorg,
155:                 (double) m yorg, (double) m_xsize, (double) m_ysize,
156:                 CDE PEN GREEN );
157:     if (vdebug) CIC_force_overlays ( oh_Console[PApp->Frame_Index] );
158: }
159:
160: void CRunLength::OnChangeSampleFactor()
161: {
162:     int temp;
163:     temp = (int) GetDlgItemInt (IDC_SAMPLE_FACTOR);
164:     if ( temp  < 5 ||  temp >= 10 )
165:     {
166:        AfxMessageBox ( " 4 < Sample < 10 이어야 합니다" );
167:        return;
168:     }
169:
170:     m sample = temp;
171:
172: }
173:
174: BOOL CRunLength::PreTranslateMessage(MSG* pMsg)
175: {
176:     // TODO: Add your specialized code here and/or call the base class
177:
178:     if (pMsg->message == WM_KEYDOWN && pMsg->wParam == 13)
179:     {
180:        if ( GetDlgItem (IDOK)->m_hWnd != pMsg->hwnd )
181:           pMsg->wParam = 9;
182:     }
183:
184:     return CDialog::PreTranslateMessage(pMsg);
185: }
186:
187: OBstruct CANOBject[130][MAX OBJECT];
188: EDGEstruct CANEdge[130][MAX OBJECT+10], XEdge[50][MAX OBJECT];
189: c Int16   OBCount[130], OBNormal[130];
190: LEAVES    Leaf[30];
191: CANLEAFOB   LeafOBject[50];
192: RECT  LeafRect[30];
193: BRANCH   BRCOOR[100];
194:
195: int xcount;
196: int  CRunLength::RunInspect ()
197: {
198:     CVt52App *PApp = (CVt52App *) AfxGetApp ();
199:
200:     // initialize the variales of RunLength Dialog
201:     m xorg  = PApp->mRunxorg;
202:     m yorg  = PApp->mRunyorg;
203:     m xsize = PApp->mRunxsize;
204:     m ysize = PApp->mRunysize;
205:     m sample = PApp->mRunsample;
206:     m diff   = PApp->mRundiff;
207:     ROIRect.left = m xorg;
208:     ROIRect.top  = m yorg;
209:     ROIRect.right  = ROIRect.left + m xsize;
210:     ROIRect.bottom = ROIRect.top + m ysize;
211:
212:     if ( PApp->SNAP () == -1 )   return -1;
213:     if ( OnRunlength1 () == -1 ) return -1;
214:     Remove Dual Edge ();
215:     if (CreateObject () == -1) return -1;
216:     ObjectMERGE ();
217:     ObjectDevide ();
218:     ObjectLINK ();
219:     OverlapOBject ();
220:     FindCrossPoint ();
221:     if (relinkflag)
222:        FindRelinkPoint ();
223:     FindIslandLeaf ();
```

```
224:     FindCuttingPoint ();
225:
226:     return 1;
227: }
228: void CRunLength::OnRunlength()
229: {
230:     OnRunlength1 ();
231: }
232:
233: int  CRunLength::OnRunlength1()
234: {
235:     // TODO: Add your control notification handler code here
236:     CPB oh oh RunLength;
237:     c Int32  flag;
238:     CVt52App *PApp = (CVt52App *) AfxGetApp ();
239:     c UInt8  prev, next, pprev, nnext, MINUSEDGE;
240:     int   diff, color, ycount, AUTOLEVEL, temp1, temp2;
241:     int pbin, nbin, ret, EDGE, grrun, grmax, grmin;
242:     int BACKLEVEL, BKENABLE, PLUSEDGE, obcount, backinit, backend;
243:     CHANGEPOINT ChangeThres [20], ChangeBuf [20];
244:     int cid, cntbuf, CONTI, PCONTI;
245:     int sum, ecount, temp3, emax, emin;
246:     int emaxcnt, emincnt, maxsum, maxcnt;
247:     int minsum, mincnt;
248:
249:     // allocate the ROI Resource
250:     oh RunLength = CPB alloc ();
251:     CPB bind ( oh RunLength, windowHandle[0], 1, &flag);
252:     //CPB bind ( oh RunLength, windowHandle[PApp->Frame Index], 1, &flag);
253:     CPB set window root ( oh RunLength, m xorg, m yorg, m xsize+1,
254:                           m ysize+1, 0, 0);
255:
256:     xcount = 0;
257:     grrun = thres;
258:     maxsum = 0;
259:     maxcnt = 0;
260:     minsum = 0;
261:     mincnt = 0;
262:     if (vdebug) CIC clear overlays ( oh Console[PApp->Frame Index] );
263:     for ( int i = 0; i < m_xsize; i += m_sample, xcount++ )
264:     {
265:        pprev = -1;
266:        ycount = 0;
267:        OBNormal [xcount] = 0;
268:        MINUSEDGE = 0;
269:        AUTOLEVEL = 0;
270:        BACKLEVEL = 0;
271:        BKENABLE = 0;
272:        PLUSEDGE = 0;
273:        temp1 = 0;
274:        temp2 = 300;
275:        obcount = 0;
276:        cid = 0;
277:        ChangeThres [0].thres = 0;
278:        PCONTI = 0;
279:        sum = 0;
280:        ecount = 0;
281:        emax = 0;
282:        emin = 300;
283:        emaxcnt = 0;
284:        emincnt = 0;
285:
286:        if ( i && grmax > 0 )
287:           grrun = grmax;
288:        else
289:           grrun = thres;
290:        grmax = 0;
291:        grmin = 300;
292:        for ( int j = 1; j < m_ysize ; j++ )
293:        {
294:           if ( i ) PCONTI = CONTI;
295:           prev = CPB pixel get (oh RunLength, i, j-1);
296:           next = CPB pixel get (oh RunLength, i, j+1);
297:           diff = next - prev;
298:           if ( diff >= 0 )
```

```
299:            {
300:               if ( prev > grrun ) pbin = 0xff;
301:               else                pbin = 0;
302:
303:               if ( next >= grrun )   nbin = 0xff;
304:               else                nbin = 0;
305:            }
306:            else
307:            {
308:               if ( prev >= grrun ) pbin = 0xff;
309:               else                pbin = 0;
310:
311:               if ( next > grrun )   nbin = 0xff;
312:               else                nbin = 0;
313:            }
314:            if ( abs ( diff ) > m_diff ) EDGE = 1;
315:            else if ( abs ( diff ) == m_diff )
316:            {
317:               if ( j > 1 && j < ( m_ysize - 1 ) )
318:               {
319:                  nnext = CPB pixel get ( oh RunLength, i, j+2 );
320:                  diff = nnext - pprev;
321:                  ret = CheckGray ( pprev, prev, next, nnext );
322:                  if ( abs (diff) > m_diff && ret == 1 ) EDGE = 1;
323:                  else EDGE =0;
324:               }
325:            }
326:            else
327:               EDGE = 0;
328:
329:            pprev = prev;
330:
331:            if ( ycount >= ( MAX_OBJECT + 10 ) )
332:            {
333:               int ret;
334:               ret = AfxMessageBox ( "Edge의 수가 초과됨", MB_OKCANCEL );
335:               if ( ret == IDCANCEL )
336:               {
337:                  // delete the resource
338:                  CPB set unbound ( oh RunLength );
339:                  CPB free (oh RunLength );
340:                  return -1;
341:               }
342:            }
343:
344:            CONTI = 0;
345:            if ( EDGE )
346:            {
347:               temp3 = CPB pixel get ( oh RunLength, i, j ) & 0xff;
348:               if ( temp3 > emax )
349:               {
350:                  emax = temp3;
351:                  emaxcnt = 1;
352:               }
353:               else if ( temp3 == emax )
354:                  emaxcnt++;
355:               if ( temp3 < emin )
356:               {
357:                  emin = temp3;
358:                  emincnt = 1;
359:               }
360:               else if ( temp3 == emin )
361:                  emincnt++;
362:               sum += temp3;
363:               ecount++;
364:            }
365:
366:            if ( EDGE == 1 && pbin != nbin )
367:            {
368:               CONTI = 1;
369:               CANEdge [xcount][ycount].edge = j;
370:               if ( diff >= 0 )
371:               {
372:                  if ( AUTOLEVEL == 1 ) AUTOLEVEL = 2;
373:                  if ( BACKLEVEL == 0 )
```

```
374:                {
375:                   backinit = j;
376:                   BACKLEVEL = 1;
377:                }
378:                else if ( BACKLEVEL == 1 && PLUSEDGE == 0 )
379:                {
380:                   backinit = j;
381:                   PLUSEDGE = 1;
382:                }
383:
384:                if ((OBNormal [xcount] != 1) && (OBNormal [xcount] != 3))
385:                   OBNormal [xcount] += 1;
386:                CANEdge[xcount][ycount++].sign = PLUS;
387:                color = CDE PEN RED;
388:             }
389:             else
390:             {
391:                if ( AUTOLEVEL == 0 )
392:                {  backend = j;
393:                   AUTOLEVEL = 1;
394:                }
395:                else if ( AUTOLEVEL == 1 && PCONTI == 0 && CONTI )
396:                   backend = j;
397:
398:                if ( BACKLEVEL == 1 ) BACKLEVEL = 2;
399:
400:                if ((OBNormal [xcount] != 2) && (OBNormal [xcount] != 3))
401:                   OBNormal [xcount] += 2;
402:                CANEdge[xcount][ycount++].sign = MINUS;
403:                color = CDE PEN BLUE;
404:             }
405:             // draw the cross hair line (application point)
406:             if (vdebug) CIC_draw_cross ( oh_Console[PApp->Frame_Index],
      (double) (i+m_xorg),
407:                                (double) (j+m_yorg), 3 , 0, color );
408:          }
409:
410:          if ( AUTOLEVEL == 1 )
411:          {
412:             if ( CONTI && PCONTI == 0 )
413:             {
414:                temp1 = 0;
415:             }
416:             if ( next > temp1 ) temp1 = next;
417:          }
418:
419:          if ( AUTOLEVEL == 2 )
420:          {
421:             ChangeThres [obcount].thres = temp1;
422:             ChangeThres [obcount++].yindex = -1;
423:             temp1 = 0;
424:             grmax = 0;
425:             AUTOLEVEL = 0;
426:             if ( BKENABLE == 0 ) BKENABLE = 1;
427:          }
428:
429:          if ( BKENABLE )
430:          {
431:             if ( BACKLEVEL == 1 )
432:             {
433:                if ( PLUSEDGE == 1 )
434:                {
435:                   temp2 = 300;
436:                   PLUSEDGE = 0;
437:                }
438:                if ( next < temp2 ) temp2 = next;
439:             }
440:             if ( BACKLEVEL == 2 )
441:             {
442:                if ( temp2 < grmin ) grmin = temp2;
443:                ChangeThres [obcount-1].yindex = (backinit + backend) >> 1;
444:                temp2 = 300;
445:                BACKLEVEL = 0;
446:             }
447:          }
```

```
448:
449:        }
450:
451:        if ( ecount && ( ecount > emincnt ) )
452:        {
453:           sum -= emin * emincnt;
454:           ecount -= emincnt;
455:           grmax = sum / ecount;
456:           if ( emax > emin )
457:           {
458:              maxsum += grmax;
459:              maxcnt++;
460:           }
461:           minsum += emin;
462:           mincnt++;
463:        }
464:        else grmax = 0;
465:        for ( j = 0 ; j < obcount ; j++ )
466:        {
467:           ChangeBuf [j].yindex = ChangeThres [j].yindex;
468:           ChangeBuf [j].thres  = ChangeThres [j].thres;
469:        }
470:        cntbuf = obcount;
471:        CANEdge[xcount][ycount].edge = -1;
472:     }
473:
474:     maxedge = maxsum / maxcnt;
475:     minedge = minsum / mincnt;
476:     if (vdebug) CIC_force_refresh  ( oh_Console[PApp->Frame_Index] );
477:
478:     // delete the resource
479:     CPB set unbound ( oh RunLength );
480:     CPB free (oh RunLength );
481:
482:     return 1;
483: }
484:
485: int  CRunLength::CheckGray ( int pprev, int prev, int next, int nnext )
486: {
487:     int diff1, diff2, diff3;
488:     int EDGE;
489:
490:     diff1 = prev - pprev;
491:     diff2 = next - prev;
492:     diff3 = nnext - next;
493:
494:     if ( (diff1 >= 0) && (diff2 >= 0) && (diff3 >= 0) )        EDGE = 1;
495:     else if ( (diff1 <= 0 ) && (diff2 <= 0) && (diff3 <= 0) ) EDGE = 1;
496:     else EDGE = 0;
497:
498:     if ( EDGE ) return 1;
499:     else        return 0;
500: }
501:
502: void CRunLength::OnChangeEdgeDiff()
503: {
504:     m_diff = (int) GetDlgItemInt (IDC_EDGE_DIFF);
505: }
506:
507: void CRunLength::Remove_Dual_Edge ()
508: {
509:     int color, xc, i, j;
510:     short csign, psign;
511:     CVt52App *PApp = (CVt52App *) AfxGetApp ();
512:     CPB oh oh RunLength;
513:     c_Int8  mono;
514:     c_Int32 flag, PLUS Count;
515:
516:     oh RunLength = CPB alloc ();
517:     CPB bind ( oh RunLength, windowHandle[0], 1, &flag);
518:     CPB set window root ( oh RunLength, m_xorg, m_yorg, m_xsize+1,
519:                           m_ysize+1, 0, 0);
520:
521:     if (vdebug) CIC_clear_overlays ( oh_Console[PApp->Frame_Index] );
522:     xc = 0;
```

```
523:     for (i = 0; i < xcount; i++, xc += m_sample)
524:     {
525:        PLUS Count = 0;
526:        for ( j = 1; (CANEdge[i][j-1].edge != -1) &&
527:                     (CANEdge[i][j].edge != -1); j++ )
528:        {
529:           psign = CANEdge[i][j-1].sign;
530:           csign = CANEdge[i][j].sign;
531:           if (psign == MINUS)
532:           {
533:              PLUS Count = 0;
534:              color = CDE PEN BLUE;
535:           }
536:           else
537:           {
538:              PLUS Count++;
539:              color = CDE PEN RED;
540:           }
541:
542:           if (psign == csign)
543:           {
544:              mono = edge type ( oh RunLength, xc, CANEdge[i][j-1].edge,
545:                                 CANEdge[i][j].edge, csign);
546:
547:              if (mono == OK)
548:              {
549:                 color = CDE PEN YELLOW;
550:                 if (PLUS_Count == 1) color = CDE_PEN_RED;
551:              }
552:              else
553:              {
554:                 if (PLUS Count > 1) color = CDE_PEN_YELLOW;
555:                 PLUS Count = 0;
556:              }
557:           }
558:           else if ( (psign == PLUS) && (csign == MINUS) )
559:           {
560:              if (PLUS_Count > 1) color = CDE_PEN_YELLOW;
561:           }
562:           // draw the cross hair line (application point)
563:           if ( color == CDE PEN YELLOW )
564:              CANEdge[i][j-1].sign = VIRTUAL;
565:           if (vdebug) CIC_draw_cross ( oh_Console[PApp->Frame_Index], (double)
    (xc+m xorg),
566:                                 (double) (CANEdge[i][j-1].edge+m_yorg), 3 , 0, color );
567:        }
568:        if (CANEdge[i][j-1].edge != -1)
569:        {
570:           psign =CANEdge[i][j-1].sign;
571:           if (psign == MINUS) color = CDE_PEN_BLUE;
572:           else
573:              if (PLUS_Count == 0) color = CDE_PEN_RED;
574:              else
575:              {
576:                 color = CDE PEN YELLOW;
577:                 CANEdge[i][j-1].sign = VIRTUAL;
578:              }
579:
580:           if (vdebug) CIC_draw_cross ( oh_Console[PApp->Frame_Index], (double)
    (xc+m xorg),
581:                                 (double) (CANEdge[i][j-1].edge+m_yorg), 3 , 0, color );
582:        }
583:
584:        int k = 0;
585:        for ( j = 0 ; CANEdge[i][j].edge != -1 ; j++ )
586:        {
587:           if ( CANEdge[i][j].sign != VIRTUAL )
588:           {
589:              CANEdge[i][k].edge   = CANEdge[i][j].edge;
590:              CANEdge[i][k++].sign = CANEdge[i][j].sign;
591:           }
592:        }
593:
594:        CANEdge[i][k].edge = -1;
595:     }
```

```
596:
597:    if (vdebug) CIC_force_refresh  ( oh_Console[PApp->Frame_Index] );
598:
599:    // delete the resource
600:    CPB set unbound ( oh RunLength );
601:    CPB free (oh RunLength );
602: }
603:
604: BOOL edge type ( CPB oh oh RunLength, int xc, int yinit, int yend, short
     dsign)
605: {
606:    c UInt8  curr, next;
607:    short isolate;
608:
609:    isolate = 0;
610:    for (int i = yinit; i < yend ; i++)
611:    {
612:       curr = CPB pixel get (oh RunLength, xc, i);
613:       next = CPB pixel get (oh RunLength, xc, i+1);
614:
615:       if ((next - curr) > 0)
616:       {
617:          if (dsign == MINUS) isolate++;
618:       }
619:       else if ((next - curr) < 0)
620:       {
621:          if (dsign == PLUS ) isolate++;
622:       }
623:       else;
624:
625:       if (isolate > 0)  return (SYSERR);
626:    }
627:
628:    return (OK);
629: }
630:
631: void CRunLength::OnRunlength2 ()
632: {
633:    Remove Dual Edge ();
634: }
635:
636: #define ILLEGAL OBJECT  120
637:
638: int  CRunLength::CreateObject ()
639: {
640:    int   REALEdge[MAX OBJECT], init, end, count, ob_no, length;
641:    short csign[MAX OBJECT], sign;
642:    CPB oh oh RunLength;
643:    c Int32 flag, xc, color, update;
644:    c UInt8  curr;
645:    c UInt16 min, max, avg, temp;
646:    double avg_length, dev_length;
647:
648:    CVt52App *PApp = (CVt52App *) AfxGetApp ();
649:    oh RunLength = CPB alloc ();
650:    CPB bind ( oh RunLength, windowHandle[0], 1, &flag);
651:    CPB set window root ( oh RunLength, m xorg, m yorg, m xsize+1,
652:                     m ysize+1, 0, 0);
653:
654:    xc = 0;
655:    leafindex = 0;
656:    for ( int i = 0 ; i < xcount ; i++, xc += m_sample)
657:    {
658:       count = 0; ob no = 0;
659:       // OBNormal [i] = 0;
660:       for (int j = 0 ; CANEdge[i][j].edge != -1 ; j++ )
661:       {
662:          sign = CANEdge[i][j].sign;
663:          if ( sign == PLUS || sign == MINUS )
664:          {
665:             REALEdge[count] = CANEdge[i][j].edge;
666:             csign[count++] = sign;
667:             if ( count >= MAX_OBJECT )
668:             {
669:                int ret;
```

```
670:                ret = AfxMessageBox ( "Object수 초과", MB_OKCANCEL );
671:                if ( ret == IDCANCEL )
672:                {
673:                   // delete the resource
674:                   CPB set unbound ( oh RunLength );
675:                   CPB free ( oh RunLength );
676:                   return -1;
677:                }
678:             }
679:          }
680:       }
681:
682:       for (j = 0 ; j < (count - 1) ; j++ )
683:       {
684:          update = 0;
685:          if ( csign[j] == PLUS
686:             && csign[j+1] == MINUS)
687:          {
688:          }
689:          else
690:          {
691:             init = REALEdge [j];
692:             end  = REALEdge [j+1];
693:             if (init >= end)
694:                temp  = 0;
695:             min = 256; max = 0; avg = 0;
696:             for (int k = init ; k <= end ; k++)
697:             {
698:                curr = CPB pixel get (oh RunLength, xc, k);
699:                if ( min > curr ) min = curr;
700:                if ( max < curr ) max = curr;
701:                avg += curr;
702:             }
703:             length = end - init + 1;
704:             CANObject[i][ob no].grsum = avg;
705:             avg /= length;
706:             if ((csign[j] == MINUS) && (csign[j+1] == PLUS))
707:             {
708:                if ( avg < ILLEGAL_OBJECT )
709:                {
710:                 CANObject[i][ob no].type = NORMAL;
711:                 update = 1;
712:                }
713:                else
714:                   update = 0;
715:             }
716:             else
717:             {
718:                if (abs(end - init) < 80)
719:                {
720:                   update = 1;
721:                   CANObject[i][ob no].type = REFINE;
722:                }
723:                else update = 0;
724:             }
725:             CANObject[i][ob no].next id = -1;
726:             CANObject[i][ob no].prev id = -1;
727:             CANObject[i][ob no].cross = 0;
728:             CANObject[i][ob no].count = 0;
729:             if (update)
730:             {
731:                color = CDE PEN GREEN;
732:                CANObject[i][ob no].fedge  = init;
733:                CANObject[i][ob no].redge  = end;
734:                CANObject[i][ob no].length = end - init + 1;
735:                CANObject[i][ob no].min = min;
736:                CANObject[i][ob no].max = max;
737:                CANObject[i][ob no].avg = avg;
738:                CANObject[i][ob no].from = csign[j];
739:                CANObject[i][ob no].to   = csign[j+1];
740:                CANObject[i][ob no].orient = 0;
741:                avg length = avg / length;
742:                dev length = (max - min) / length;
743:
744:                if ( CANObject[i][ob_no].length > LEAFLENGTH )
```

```
745:                {
746:                   color = CDE_PEN_RED;
747:                   CANOBject[i][ob_no].leaf = 1;
748:                   LeafOBject[leafindex].xindex = i;
749:                   LeafOBject[leafindex++].yindex = ob_no;
750:                }
751:                else
752:                   CANOBject[i][ob_no].leaf = 0;
753:
754:                ob_no++;
755:                if (vdebug) CIC_draw_line ( oh_Console[PApp->Frame_Index],
756:                      (double) (xc+m_xorg), (double) (init+m_yorg), (double)
    (xc+m_xorg),
757:                      (double) (end+m_yorg), color );
758:             }
759:
760:          }
761:       }
762:
763:       // Object No. per line
764:       OBCount[i] = ob_no;
765:    }
766:
767:    if (vdebug) CIC_force_refresh  ( oh_Console[PApp->Frame_Index] );
768:
769:    // delete the resource
770:    CPB_set_unbound ( oh_RunLength );
771:    CPB_free ( oh_RunLength );
772:
773:    return 1;
774: }
775:
776: void CRunLength::OnRunlength3()
777: {
778:    // TODO: Add your control notification handler code here
779:    CVt52App *PApp = (CVt52App *) AfxGetApp();
780:
781:    CreateObject ();
782:    ObjectMERGE ();
783:    ObjectDevide ();
784:    ObjectLINK ();
785:    OverlapOBject ();
786:    FindCrossPoint ();
787:    if (relinkflag)
788:       FindRelinkPoint ();
789:    FindIslandLeaf ();
790:    FindCuttingPoint ();
791:    PApp->ReportCutPoints ();
792: }
793:
794: void CRunLength::ObjectFill ()
795: {
796:    int xinit, xend, xc;
797:    int diff, ycount;
798:    c_Int32 flag;
799:    CPB oh oh_RunLength;
800:    c_UInt8  prev, next;
801:    int color, level;
802:    CVt52App *PApp = (CVt52App *) AfxGetApp ();
803:
804:    // find the initial valid object
805:    for ( int i = 0 ; i < (xcount-1) ; i++ )
806:    {
807:       if ( (OBNormal[i] == 3) && (OBNormal[i+1] == 3) &&
808:            (OBNormal[i+2] == 3) )
809:       {
810:          xinit = i;
811:          break;
812:       }
813:    }
814:
815:    // find the tail object
816:    for ( i = xcount - 1 ; i >= 0 ; i--)
817:    {
818:       if ( (OBNormal[i] == 3) && (OBNormal[i-1] == 3) &&
```

```
819:             (OBNormal[i-2] == 3) )
820:         {
821:            xend = i;
822:            break;
823:         }
824:      }
825:
826:      // illegal condition
827:      if ( xinit >= xend ) return;
828:
829:      oh RunLength = CPB alloc ();
830:      CPB bind ( oh RunLength, windowHandle[0], 1, &flag);
831:      CPB set window root ( oh RunLength, m xorg, m yorg, m xsize+1,
832:                            m ysize+1, 0, 0);
833:      // fill empty object
834:      xc = xinit * m sample;
835:      for (i = xinit; i <= xend; i++, xc += m_sample)
836:      {
837:         if (OBNormal [i] != 3)
838:         {
839:            if ( m diff > 8)
840:               level = m diff - ( m diff - 8) / 2;
841:            else
842:               level = m diff - 1;
843:            for ( int k = 0; k < 2; k++ ) {
844:               OBNormal [i] = 0;
845:               ycount = 0;
846:               for ( int j = 1; j < m_ysize ; j++)
847:               {
848:                  prev = CPB pixel get (oh RunLength, xc, j-1);
849:                  next = CPB pixel get (oh RunLength, xc, j+1);
850:                  diff = next - prev;
851:
852:                  if ( abs(diff) > level )
853:                  {
854:                     CANEdge[i][ycount].edge = j;
855:                     if ( diff >= 0 )
856:                     {
857:                        if ((OBNormal [i] != 1) && (OBNormal [i] != 3))
858:                           OBNormal [i] += 1;
859:                        CANEdge[i][ycount++].sign = PLUS;
860:                        color = CDE PEN ORANGE;
861:                     }
862:                     else
863:                     {
864:                        if ((OBNormal [i] != 2) && (OBNormal [i] != 3))
865:                              OBNormal [i] += 2;
866:                        CANEdge[i][ycount++].sign = MINUS;
867:                        color = CDE PEN MAGENTA;
868:                     }
869:
870:                     // draw the cross hair line (application point)
871:                     if (vdebug) CIC_draw_cross ( oh_Console[PApp->Frame_Index],
    (double) (xc+m xorg),
872:                                 (double) (j+m_yorg), 3 , 0, color );
873:
874:                  }
875:               }
876:               if (OBNormal [i] == 3) {
877:                  CANEdge[i][ycount].edge = -1;
878:                  break;
879:               }
880:               if ( m diff > 8)
881:                  level = level - ( m diff - 8) / 2;
882:               else
883:                  level -= 1;
884:            }
885:         }
886:      }
887:      if (vdebug) CIC_force_refresh  ( oh_Console[PApp->Frame_Index] );
888:
889:      // delete the resource
890:      CPB set unbound ( oh RunLength );
891:      CPB free (oh RunLength );
892: }
```

```
893:
894: void CRunLength::ObjectMERGE()
895: {
896:     short cmin, cmax, pmin, pmax, nmin, nmax, min, max;
897:     short cavg, navg, pavg;
898:     int       edge;
899:
900:     for ( int i = 0 ; i < xcount ; i++ )
901:     {
902:        for ( int j = 0 ; j < OBCount [i] ; j++ )
903:        {
904:           if ( CANObject [i][j].type == REFINE )
905:           {
906:              if ( CANObject[i][j].from == MINUS ) edge = -1;
907:              else                                 edge =  1;
908:
909:              if ( (edge == -1) && (j <  (OBCount [i] - 1) ) &&
910:                   (CANObject[i][j+1].type == NORMAL) &&
911:                   (CANObject[i][j].redge == CANObject[i][j+1].fedge) )
912:              {
913:                 cavg = CANObject[i][j].avg;
914:                 cmin = CANObject[i][j].min;
915:                 cmax = CANObject[i][j].max;
916:
917:                 navg = CANObject[i][j+1].avg;
918:                 nmin = CANObject[i][j+1].min;
919:                 nmax = CANObject[i][j+1].max;
920:
921:                 if ( cmin < nmin ) min = cmin;
922:                 else               min = nmin;
923:
924:                 if ( cmax > nmax ) max = cmax;
925:                 else               max = nmax;
926:                 if ( abs(max - min) <= 16 )
927:                    OBjectSort ( i, j+1, j, 1);
928:              }
929:              else if ( (edge == 1) && (j != 0) && (CANObject[i][j-1].type ==
     NORMAL)
930:                 && (CANObject[i][j].fedge == CANObject[i][j-1].redge) )
931:              {
932:                 cavg = CANObject[i][j].avg;
933:                 cmin = CANObject[i][j].min;
934:                 cmax = CANObject[i][j].max;
935:
936:                 pavg = CANObject[i][j-1].avg;
937:                 pmin = CANObject[i][j-1].min;
938:                 pmax = CANObject[i][j-1].max;
939:
940:                 if ( cmin < pmin ) min = cmin;
941:                 else               min = pmin;
942:
943:                 if ( cmax > pmax ) max = cmax;
944:                 else               max = pmax;
945:                 if ( abs(max - min) <= 16 )
946:                    OBjectSort ( i, j-1, j, 1);
947:              }
948:           }
949:        }
950:     }
951: }
952:
953: void CRunLength::ObjectDevide ()
954: {
955:     int cob no, nob no, angle;
956:     int diff, min, pos, cavg, navg, ynpos;
957:     int cindex, nindex, prev id;
958:     int COB ID [MAX OBJECT], NOB ID [MAX OBJECT];
959:     CVt52App *PApp = (CVt52App *) AfxGetApp ();
960:
961:     angle = 40;
962:     for ( int i = 0 ; i < (xcount-1) ; i++ )
963:     {
964:        cob no = 0; nob no = 0;
965:        for ( int j = 0 ; j < OBCount [i] ; j++ )
966:           if ( CANObject [i][j].type == NORMAL )
```

```
 967:             COB ID [cob no++] = j;
 968:
 969:       for ( j = 0; j < OBCount[i+1] ; j++ )
 970:          if ( CANObject [i+1][j].type == NORMAL )
 971:             NOB ID [nob no++] = j;
 972:
 973:       if ( cob_no && nob_no )
 974:       {
 975:          for ( j = 0 ; j < cob_no ; j++ )
 976:          {
 977:             min = 0x0ff0;
 978:             cindex = COB ID [j];
 979:             pos = 0;
 980:
 981:             cavg = CANObject[i][cindex].redge +
 982:                  CANObject[i][cindex].fedge;
 983:
 984:             for ( int k = 0 ; k < nob_no ; k++ )
 985:             {
 986:                nindex = NOB ID [k];
 987:                navg = CANObject[i+1][nindex].redge +
 988:                     CANObject[i+1][nindex].fedge;
 989:                diff = navg - cavg;
 990:
 991:                if (diff < 0) diff = -diff;
 992:
 993:                // add decision rule
 994:                if (min > diff)
 995:                {
 996:                   min = diff;
 997:                   pos = nindex;
 998:                   ynpos = navg;
 999:                }
1000:             }
1001:
1002:             if (abs(ynpos - cavg) < angle)
1003:             {
1004:                int pfedge, predge, newid;
1005:                newid = 0;
1006:                prev id = CANObject [i][cindex].prev id;
1007:                CANObject [i][cindex].next id = pos;
1008:                if (prev_id != -1)
1009:                {
1010:                   pfedge = CANObject[i-1][prev id].fedge;
1011:                   predge = CANObject[i-1][prev id].redge;
1012:                }
1013:                int clength, nlength;
1014:
1015:                clength = CANObject [i][cindex].length;
1016:                nlength = CANObject [i+1][pos].length;
1017:
1018:                if ( (nlength - clength ) >= clength )
1019:                {
1020:                   int cfedge, credge, nfedge, nredge;
1021:                   int efedge, eredge;
1022:
1023:                   cfedge = CANObject [i][cindex].fedge;
1024:                   credge = CANObject [i][cindex].redge;
1025:                   nfedge = CANObject [i+1][pos].fedge;
1026:                   nredge = CANObject [i+1][pos].redge;
1027:
1028:                   if ( credge > nfedge && cfedge < nredge )
1029:                   {
1030:                      //if ( prev_id != -1)
1031:                      //{
1032:                      if ( abs ( cfedge - nfedge ) <= 1 )
1033:                         newid = OBjectSplit ( i, cindex, pos, 0 );
1034:                      else if ( abs ( credge - nredge ) <= 1 )
1035:                         newid = OBjectSplit ( i, cindex, pos, 1 );
1036:                      if ( newid == 0 )
1037:                      {
1038:                         efedge = cfedge + cfedge - pfedge;
1039:                         eredge = credge + credge - predge;
1040:                         if ( abs ( efedge - nfedge ) <= 1 ||
1041:                              abs ( cfedge - nfedge ) <= 1 )
```

```
1042:                              newid = OBjectSplit ( i, cindex, pos, 0 );
1043:                           else if ( abs ( eredge - nredge ) <= 1 ||
1044:                              abs ( credge - nredge ) <= 1 )
1045:                              newid = OBjectSplit ( i, cindex, pos, 1 );
1046:                        }
1047:                     }
1048:                  }
1049:                  if ( CANOBject [i+1][pos].prev id == -1 )
1050:                      CANOBject [i+1][pos].prev id = cindex;
1051:                  if ( newid )
1052:                      CANOBject [i+1][newid].prev id = cindex;
1053:               }
1054:            }
1055:         }
1056:      }
1057:
1058:      if (vdebug) CIC_force_refresh  ( oh_Console[PApp->Frame_Index] );
1059:   }
1060:   void CRunLength::ObjectLINK ()
1061:   {
1062:      int cob no, nob no, count, maxpos, temp;
1063:      int COB ID [MAX OBJECT], NOB ID [MAX_OBJECT];
1064:      int prev id, next_id, maxcount;
1065:      c Int32 flag;
1066:      CPB oh oh RunLength;
1067:      CVt52App *PApp = (CVt52App *) AfxGetApp ();
1068:
1069:      oh RunLength = CPB alloc ();
1070:      CPB bind ( oh RunLength, windowHandle[0], 1, &flag);
1071:      CPB set window root ( oh RunLength, m xorg, m yorg, m xsize+1,
1072:                            m ysize+1, 0, 0);
1073:
1074:      brcount = 0;
1075:      maxcount = 0;
1076:      relinkflag = 0;
1077:      for ( int i = 0 ; i < (xcount-1) ; i++ )
1078:      {
1079:         cob no = 0; nob no = 0; count = 0; maxpos = -1;
1080:         for ( int j = 0 ; j < OBCount [i] ; j++ )
1081:         {
1082:            if ( i == 0 )
1083:            {
1084:               CANOBject [i][j].prev id = -1;
1085:               CANOBject [i][j].next id = -1;
1086:            }
1087:
1088:            if ( CANOBject [i][j].type == NORMAL )
1089:            {
1090:               if ( CANOBject [i][j].cross == OBCHECK )
1091:               {
1092:                  prev id = CANOBject [i][j].prev id;
1093:                  if ( prev id == -1 )
1094:                        CANOBject [i][j].cross = OBCHILD;
1095:                  else
1096:                  {
1097:                     temp = CANOBject [i-1][prev id].count;
1098:                     if ( temp > count )
1099:                     {
1100:                        CANOBject [i][j].cross = OBPARENT;
1101:                        if ( maxpos != -1 )
1102:                           CANOBject [i][maxpos].cross = OBCHILD;
1103:                        count = temp;
1104:                        maxpos = j;
1105:                     }
1106:                  }
1107:               }
1108:               COB ID [cob no++] = j;
1109:            }
1110:         }
1111:
1112:         for ( j = 0; j < OBCount[i+1] ; j++)
1113:         {
1114:            CANOBject [i+1][j].prev id = -1;
1115:            CANOBject [i+1][j].next id = -1;
1116:
```

```
1117:            if ( CANObject [i+1][j].type == NORMAL )
1118:               NOB ID [nob no++] = j;
1119:         }
1120:         if ( cob_no && nob_no )
1121:         {
1122:            MakeChain (oh RunLength, i, COB ID, NOB ID, cob no, nob no, 0, 0);
1123:         }
1124:
1125:         for ( j = 0 ; j < OBCount [i] ; j++ )
1126:         {
1127:            next id = CANObject [i][j].next id;
1128:            if ( next_id == -1 )
1129:            {
1130:               prev id = CANObject [i][j].prev id;
1131:                if ( prev_id != -1 )
1132:               {
1133:                  count = CANObject [i-1][prev id].count + 1;
1134:                   CANObject [i][j].count = count;
1135:                  if (count > maxcount)
1136:                  {
1137:                     relinkflag = 1;
1138:                     maxcount = count;
1139:                     relinkindex = i;
1140:                     relinkid = j;
1141:                  }
1142:                  BRCOOR [brcount].xid = i;
1143:                  BRCOOR [brcount++].yid = j;
1144:               }
1145:            }
1146:         }
1147:
1148:      }
1149:
1150:      BRCOOR[brcount].xid = -1;
1151:
1152:         // delete the resource
1153:      CPB set unbound ( oh RunLength );
1154:      CPB free (oh RunLength );
1155:
1156:      if (vdebug) CIC_force_refresh  ( oh_Console[PApp->Frame_Index] );
1157: }
1158:
1159: void CRunLength::MakeChain ( CPB_oh oh_RunLength, int i, int *frchain, int
     *tochain,
1160:                int cob_no, int nob_no, int change, int next )
1161: {
1162:      int prev id, pfedge, predge, pcent, estcent, estlength;
1163:      int xc, pos, cindex, nindex, angle, grpos, grynpos;
1164:      int cavg, navg, ynpos, xnext, cgravg, ngravg;
1165:      CVt52App *PApp = (CVt52App *) AfxGetApp ();
1166:      c UInt16 min, grmin, kkk;
1167:      c Int16   diff, grdiff;
1168:      int   clength, nlength;
1169:
1170:      // make the connection between the objects
1171:      xc = i *  m sample;
1172:      if (next)
1173:      {
1174:         xnext = i + 2;
1175:         angle = 70;
1176:      }
1177:      else
1178:      {
1179:         xnext = i + 1;
1180:         angle = 40;
1181:      }
1182:      for ( int j = 0 ; j < cob_no ; j++ )
1183:      {
1184:         min = grmin = 0x0ff0;
1185:         cindex = *(frchain + j);
1186:         pos = 0;
1187:         if (change)
1188:         {
1189:            cavg = CANObject[xnext][cindex].redge +
1190:                  CANObject[xnext][cindex].fedge;
```

```
1191:            cgravg  = CANObject[xnext][cindex].avg;
1192:         }
1193:         else
1194:         {
1195:            cavg = CANObject[i][cindex].redge +
1196:                  CANObject[i][cindex].fedge;
1197:            cgravg  = CANObject[i][cindex].avg;
1198:         }
1199:
1200:         for ( int k = 0 ; k < nob_no ; k++ )
1201:         {
1202:            nindex = *(tochain + k);
1203:            if (change)
1204:            {
1205:               navg = CANObject[i][nindex].redge +
1206:                     CANObject[i][nindex].fedge;
1207:               ngravg = CANObject[i][nindex].avg;
1208:            }
1209:            else
1210:            {
1211:               navg = CANObject[xnext][nindex].redge +
1212:                     CANObject[xnext][nindex].fedge;
1213:               ngravg = CANObject[xnext][nindex].avg;
1214:            }
1215:            diff = navg - cavg;
1216:            grdiff = ngravg - cgravg;
1217:            if (diff < 0) diff = -diff;
1218:            if (grdiff < 0) grdiff = -grdiff;
1219:
1220:            if (min > diff)
1221:            {
1222:               min = diff;
1223:               pos = nindex;
1224:               ynpos = navg;
1225:               kkk = k;
1226:            }
1227:            if (grmin > grdiff)
1228:            {
1229:               grmin = grdiff;
1230:               grpos = nindex;
1231:               grynpos = navg;
1232:               kkk = k;
1233:            }
1234:         }
1235:
1236:         int orient = 0;
1237:         diff = ynpos - cavg;
1238:
1239:         if (abs(diff) < angle)
1240:         {
1241:            int cid, nid, newid, pcount, ret;
1242:            newid = 0;
1243:            if (change)
1244:            {
1245:               cid = pos;
1246:               nid = cindex;
1247:               prev id    = CANObject[i][pos].prev id;
1248:               estcent    = ynpos;
1249:               estlength = CANObject[i][pos].length;
1250:               clength = estlength;
1251:               nlength = CANObject [i+1][cindex].length;
1252:            }
1253:            else
1254:            {
1255:               cid = cindex;
1256:               nid = pos;
1257:               prev id    = CANObject[i][cindex].prev id;
1258:               estcent    = cavg;
1259:               estlength = CANObject[i][cindex].length;
1260:               clength = estlength;
1261:               nlength = CANObject [i+1][pos].length;
1262:            }
1263:
1264:            if (prev_id != -1)
1265:            {
```

```
1266:             pfedge = CANObject[i-1][prev id].fedge;
1267:             predge = CANObject[i-1][prev id].redge;
1268:             pcent  = pfedge + predge;
1269:             pcount = CANObject[i-1][prev id].count + 1;
1270:          }
1271:          else
1272:             pcount = 0;
1273:
1274:          if ( abs ( nlength - clength ) >= clength )
1275:          {
1276:             int cfedge, credge, nfedge, nredge;
1277:             int efedge, eredge;
1278:
1279:             cfedge = CANObject [i][cid].fedge;
1280:             credge = CANObject [i][cid].redge;
1281:             nfedge = CANObject [i+1][nid].fedge;
1282:             nredge = CANObject [i+1][nid].redge;
1283:
1284:             if ( credge > nfedge && cfedge < nredge )
1285:             {
1286:                efedge = cfedge + cfedge - pfedge;
1287:                eredge = credge + credge - predge;
1288:             }
1289:          }
1290:          cavg = CANObject[i][cid].fedge +
1291:                 CANObject[i][cid].redge;
1292:          ynpos = CANObject[i+1][nid].fedge +
1293:                  CANObject[i+1][nid].redge;
1294:
1295:          ret = 0;
1296:          if ( abs (diff) > 10 )
1297:             ret = CheckValidLink ( oh RunLength, i, cavg >> 1, ynpos >> 1 );
1298:
1299:          if ( ret != -1 )
1300:          {
1301:             if (change)
1302:             {
1303:                if ( newid ) CANObject[i+1][newid].prev_id = pos;
1304:                if (next) cindex |= NEXT EMPTY FLAG;
1305:                *(tochain + kkk) |= LINK COMPLETE;
1306:             }
1307:             else
1308:             {
1309:                if ( newid ) CANObject[i+1][newid].prev_id = cindex;
1310:                if (next) pos |= NEXT EMPTY FLAG;
1311:                *(frchain + j) |= LINK COMPLETE;
1312:             }
1313:
1314:             CANObject[i][cid].orient  = orient;
1315:
1316:             prev id = CANObject[i+1][nid].prev id;
1317:             if ( prev id != -1  && prev id != cid &&
1318:                  CANObject [i][prev id].count > pcount )
1319:             {
1320:                CANObject[i][cid].next id = nid;
1321:                CANObject[i][cid].count   = pcount;
1322:             }
1323:             else
1324:             {
1325:                CANObject[i+1][nid].prev id = cid;
1326:                CANObject[i][cid].next id   = nid;
1327:                CANObject[i][cid].count     = pcount;
1328:             }
1329:
1330:             if (next) diff = xc + m xorg + (m sample << 1);
1331:             else      diff = xc + m_xorg + m_sample;
1332:
1333:             if (vdebug)
1334:                CIC draw line ( oh Console[PApp->Frame Index],
1335:                   (double) (xc+m xorg),
1336:                   (double) ((cavg>>1)+m yorg), (double) (diff),
1337:                   (double) ((ynpos>>1)+m_yorg), CDE_PEN_GRAY );
1338:          }
1339:       }
1340:    }
```

```
1341: }
1342:
1343: int  CRunLength::CheckValidLink ( CPB_oh oh_RunLength, int i, int ccent, int
      ncent )
1344: {
1345:    int ystep, xc, yc;
1346:    int gray, j, count, level;
1347:
1348:    ystep = ( ncent - ccent );
1349:
1350:    xc = i * m sample;
1351:    yc = ccent;
1352:    count = 0;
1353:    level = maxedge + 8;
1354:    for ( j = 1 ; j < 5 ; j++, xc++ )
1355:    {
1356:       yc = ccent + ystep * j / 5;
1357:       gray = CPB pixel get ( oh RunLength, xc, yc );
1358:
1359:       if ( gray > level )
1360:          count++;
1361:    }
1362:
1363:    if ( count >= 2 ) return -1;
1364:    else      return 0;
1365: }
1366:
1367: int  CRunLength::OBjectSplit ( int i, int cid, int nid, int same )
1368: {
1369:    int j, cdiff, ndiff, gray, sum, max, min;
1370:    int cfedge, credge, xc;
1371:    int nfedge, nredge, mid;
1372:    c Int32  flag;
1373:    CPB oh oh RunLength;
1374:    CVt52App *PApp = (CVt52App *) AfxGetApp ();
1375:
1376:    gray = 0;
1377:    sum  = 0;
1378:    max  = 0;
1379:    min  = 1000;
1380:    xc =  ( i + 1 ) * m sample;
1381:
1382:    cfedge = CANOBject [i][cid].fedge;
1383:    credge = CANOBject [i][cid].redge;
1384:    if ( same <= 1 )
1385:    {
1386:       nfedge = CANOBject [i+1][nid].fedge;
1387:       nredge = CANOBject [i+1][nid].redge;
1388:    }
1389:
1390:    oh RunLength = CPB alloc ();
1391:    CPB bind ( oh RunLength, windowHandle[0], 1, &flag);
1392:    //CPB bind ( oh RunLength, windowHandle[PApp->Frame Index], 1, &flag);
1393:    CPB set window root ( oh RunLength, m xorg, m yorg, m xsize+1,
1394:                          m ysize+1, 0, 0);
1395:    if ( same == 0 )
1396:    {
1397:       cdiff = credge - cfedge;
1398:       ndiff = nredge - nfedge;
1399:       for ( j = 0 ;; j++ )
1400:       {
1401:          gray = CPB pixel get ( oh RunLength, xc, nfedge+j );
1402:          sum += gray;
1403:          if ( max < gray ) max = gray;
1404:          if ( min > gray ) min = gray;
1405:
1406:          if ( j > ndiff ) break;
1407:          else if ( j == ndiff )
1408:             ObjectInit ( i+1, max, min, mid+1, nredge, sum, -1, 0 );
1409:          else if ( j == cdiff )
1410:          {
1411:             mid = nfedge + j;
1412:             ObjectInit ( i+1, max, min, nfedge, mid, sum, nid, 0 );
1413:             //ObjectInit ( i+1, max, min, nfedge, credge, sum, nid, 0 );
1414:             max = 0; min = 1000; sum = 0;
```

```
1415:            }
1416:         }
1417:         if (vdebug)
1418:         {
1419:            CIC draw cross ( oh Console[PApp->Frame Index];
1420:               (double) (xc + m xorg), (double) (mid+m_yorg),
1421:               3 , 0, CDE PEN ORANGE );
1422:            CIC draw cross ( oh Console[PApp->Frame Index];
1423:               (double) (xc + m xorg), (double) (mid+m_yorg+1),
1424:               3 , 0, CDE PEN BLUE );
1425:         }
1426:      }
1427:      else if ( same == 1 )
1428:      {
1429:         cdiff = credge - cfedge;
1430:         ndiff = nredge - nfedge;
1431:         cdiff = ndiff - cdiff;
1432:         for ( j = 0 ;; j++ )
1433:         {
1434:            gray = CPB pixel get ( oh RunLength, xc, nfedge+j );
1435:            sum += gray;
1436:            if ( max < gray ) max = gray;
1437:            if ( min > gray ) min = gray;
1438:
1439:            if ( j > ndiff ) break;
1440:            else if ( j == ndiff )
1441:               ObjectInit ( i+1, max, min, mid, nredge, sum, nid, 0 );
1442:            else if ( j == cdiff )
1443:            {
1444:               mid = nfedge + j;
1445:               ObjectInit ( i+1, max, min, nfedge, mid, sum, -1 , 0 );
1446:               //ObjectInit ( i+1, max, min, nfedge, cfedge-1, sum, -1 , 0 );
1447:               max = gray; min = gray; sum = gray;
1448:            }
1449:         }
1450:
1451:         if (vdebug)
1452:         {
1453:            CIC draw cross ( oh Console[PApp->Frame Index],
1454:               (double) (xc + m xorg),
1455:               (double) (mid+m yorg), 3 , 0, CDE PEN ORANGE );
1456:            CIC draw cross ( oh Console[PApp->Frame Index],
1457:               (double) (xc + m xorg),
1458:               (double) (mid+m_yorg+1), 3 , 0, CDE_PEN_BLUE );
1459:         }
1460:      }
1461:      else
1462:      {
1463:         cdiff = nid - cfedge;
1464:         ndiff = credge - cfedge;
1465:         for ( j = 0 ;; j++ )
1466:         {
1467:            gray = CPB pixel get ( oh RunLength, xc, cfedge+j );
1468:            sum += gray;
1469:            if ( max < gray ) max = gray;
1470:            if ( min > gray ) min = gray;
1471:
1472:            if ( j > ndiff ) break;
1473:            else if ( j == ndiff )
1474:            {
1475:               if ( same == 2 )
1476:                  ObjectInit ( i, max, min, nid, credge, sum, cid, 1 );
1477:               else
1478:                  ObjectInit ( i, max, min, nid, credge, sum, -1, 1 );
1479:            }
1480:            else if ( j == cdiff )
1481:            {
1482:               if ( same == 2 )
1483:                  ObjectInit ( i, max, min, cfedge, nid, sum, -1, 1 );
1484:               else
1485:                  ObjectInit ( i, max, min, cfedge, nid, sum, cid, 1 );
1486:               max = 0; min = 1000; sum = 0;
1487:            }
1488:         }
1489:
```

```
1490:        if (vdebug)
1491:        {
1492:           CIC draw cross ( oh Console[PApp->Frame Index],
1493:              (double) (xc + m xorg - m sample ),
1494:              (double) (nid+m yorg), 3 , 0, CDE PEN ORANGE );
1495:           CIC draw cross ( oh Console[PApp->Frame Index],
1496:              (double) (xc + m xorg - m sample),
1497:              (double) (nid+m_yorg+1), 3 , 0, CDE_PEN_BLUE );
1498:        }
1499:
1500:     }
1501:     //delete the resource
1502:     CPB set unbound ( oh RunLength );
1503:     CPB free (oh RunLength );
1504:
1505:     if ( same <= 1 ) return (OBCount[i+1] - 1);
1506:     else             return (OBCount[i] - 1);
1507: }
1508:
1509: void CRunLength::ObjectInit ( int index, int min, int max,
1510:                  int fedge, int redge, int sum, int oid, int later )
1511: {
1512:     int length;
1513:     double avg_length, dev_length;
1514:
1515:     length = redge - fedge + 1;
1516:
1517:     if ( later )
1518:     {
1519:        if ( oid == -1 )
1520:        {
1521:           oid = OBCount [index];
1522:           OBCount [index] += 1;
1523:           CANOBject [index][oid].type  = NORMAL;
1524:           CANOBject [index][oid].cross = OBCHILD;
1525:        }
1526:        else
1527:           CANOBject [index][oid].cross = OBPARENT;
1528:     }
1529:     else
1530:     {
1531:        if ( oid == -1 )
1532:        {
1533:           oid = OBCount [index];
1534:           OBCount [index] += 1;
1535:           CANOBject [index][oid].type  = NORMAL;
1536:           CANOBject [index][oid].cross = OBCHECK;
1537:        }
1538:        else
1539:           CANOBject [index][oid].cross = OBCHECK;
1540:
1541:        CANOBject [index][oid].next id = -1;
1542:        CANOBject [index][oid].prev id = -1;
1543:     }
1544:
1545:     CANOBject [index][oid].max = max;
1546:     CANOBject [index][oid].min = min;
1547:     CANOBject [index][oid].grsum = sum;
1548:     CANOBject [index][oid].fedge = fedge;
1549:     CANOBject [index][oid].redge = redge;
1550:     CANOBject [index][oid].length = length;
1551:     CANOBject [index][oid].avg = sum / length;
1552:     CANOBject [index][oid].from = MINUS;
1553:     CANOBject [index][oid].to   = PLUS;
1554:     //CANOBject [index][oid].cross = 0;
1555:     CANOBject [index][oid].count = 0;
1556:
1557:     avg length = CANOBject [index][oid].avg / length;
1558:     dev length = (max - min) / length;
1559:
1560:     if ( later )
1561:     {
1562:        if ( oid == -1 )
1563:        {
1564:           CANOBject [index][oid].leaf = 1;
```

```
1565:            LeafOBject[leafindex].xindex = index;
1566:            LeafOBject[leafindex++].yindex = oid;
1567:         }
1568:      }
1569:      else
1570:      {
1571:         if ( CANOBject[index][oid].length > LEAFLENGTH )
1572:         {
1573:            CANOBject [index][oid].leaf = 1;
1574:            LeafOBject[leafindex].xindex = index;
1575:            LeafOBject[leafindex++].yindex = oid;
1576:         }
1577:         else
1578:            CANOBject [index][oid].leaf = 0;
1579:      }
1580: }
1581:
1582: void CRunLength::Decide_Rule ( int index, int *cid, int estcent, int
      estlength )
1583: {
1584:    int AVG[6], LENGTH[6], OBNO;
1585:    int DECIDE[6];
1586:
1587:    for (int i = 0; i < 6;i++)
1588:       DECIDE[i] = 0;
1589:
1590:    AVG[1] = CANOBject[index][*cid].fedge +
1591:           CANOBject[index][*cid].redge;
1592:    LENGTH[1] = CANOBject[index][*cid].length;
1593:
1594:    OBNO = 0;
1595:    if ( CANOBject[index][*cid-1].type == REFINE &&
1596:        CANOBject[index][*cid-1].to == MINUS )
1597:    {
1598:       OBNO += 1;
1599:       AVG[0] = -1;
1600:       LENGTH[0] = -1;
1601:       AVG[3] = CANOBject[index][*cid-1].fedge +
1602:              CANOBject[index][*cid].redge;
1603:       LENGTH[3] = AVG[3] - (CANOBject[index][*cid-1].fedge << 1);
1604:    }
1605:    else
1606:    {
1607:       AVG[0] = -1;
1608:       LENGTH[0] = -1;
1609:       AVG[3] = -1;
1610:       LENGTH[3] = -1;
1611:    }
1612:
1613:    if ( CANOBject[index][*cid+1].type == REFINE &&
1614:        CANOBject[index][*cid+1].from == PLUS )
1615:    {
1616:       OBNO += 2;
1617:       AVG[2] = -1;
1618:       LENGTH[2] = -1;
1619:       AVG[4] = CANOBject[index][*cid].fedge +
1620:              CANOBject[index][*cid+1].redge;
1621:       LENGTH[4] = AVG[4] - (CANOBject[index][*cid].fedge << 1);
1622:    }
1623:    else
1624:    {
1625:       AVG[2] = -1;
1626:       LENGTH[2] = -1;
1627:       AVG[4] = -1;
1628:       LENGTH[4] = -1;
1629:    }
1630:
1631:    if (OBNO == 3)
1632:    {
1633:       AVG[5] = -1;
1634:       LENGTH[5] = -1;
1635:    }
1636:    else
1637:    {  AVG[5] = -1;
1638:       LENGTH[5] = -1;
```

```
1639:     }
1640:
1641:     for ( i = 0 ; i < 6 ; i++ )
1642:        if ( AVG[i] == -1 )
1643:           DECIDE[i] = -1;
1644:        else
1645:           DECIDE[i] += abs(LENGTH[i] - estlength) +
1646:                        abs(estcent - AVG[i]);
1647:     int min = 0xff0;
1648:     int minpos = 0;
1649:     for (i = 0 ; i < 6 ; i++ )
1650:        if (DECIDE[i] != -1)
1651:           if (min > DECIDE[i])
1652:           {
1653:              min = DECIDE[i];
1654:              minpos = i;
1655:           }
1656:
1657:     switch (minpos)
1658:     {
1659:        case 0:
1660:           CANObject[index][*cid-1].type = NORMAL;
1661:           *cid -= 1;
1662:           break;
1663:        case 2:
1664:           CANObject[index][*cid+1].type = NORMAL;
1665:           *cid += 1;
1666:           break;
1667:        case 3:
1668:           OBjectSort ( index, *cid, *cid-1, 0 );
1669:           break;
1670:        case 4:
1671:           OBjectSort ( index, *cid, *cid+1, 0 );
1672:           break;
1673:        case 5:
1674:           OBjectSort ( index, *cid, *cid-1, 0 );
1675:           OBjectSort ( index, *cid, *cid+1, 0 );
1676:           break;
1677:     }
1678: }
1679: void CRunLength::OBjectSort ( int index, int id1, int id2, int flag )
1680: {
1681:     int cmin, destmin, cmax, destmax;
1682:     int csum, destsum, clength, destlength;
1683:     int cfedge, credge, destfedge, destredge;
1684:     double avg_length, dev_length;
1685:
1686:     cmin = CANObject[index][id1].min;
1687:     cmax = CANObject[index][id1].max;
1688:     csum = CANObject[index][id1].grsum;
1689:     cfedge = CANObject[index][id1].fedge;
1690:     credge = CANObject[index][id1].redge;
1691:     clength = CANObject[index][id1].length;
1692:
1693:     destmin = CANObject[index][id2].min;
1694:     destmax = CANObject[index][id2].max;
1695:     destsum = CANObject[index][id2].grsum;
1696:     destfedge = CANObject[index][id2].fedge;
1697:     destredge = CANObject[index][id2].redge;
1698:     destlength = CANObject[index][id2].length;
1699:
1700:     if (cmin > destmin)
1701:     {
1702:        cmin = destmin;
1703:        CANObject[index][id1].min = destmin;
1704:     }
1705:     if (cmax < destmax)
1706:     {
1707:        cmax = destmax;
1708:        CANObject[index][id1].max = destmax;
1709:     }
1710:     if (cfedge > destfedge)
1711:        CANObject[index][id1].fedge = destfedge;
1712:     if (credge < destredge)
1713:        CANObject[index][id1].redge = destredge;
```

```
1714:
1715:     CANOBject[index][id1].grsum = csum + destsum;
1716:     CANOBject[index][id1].length = clength + destlength - 1;
1717:     CANOBject[index][id1].avg = (csum + destsum) /
1718:        CANOBject[index][id1].length;
1719:     CANOBject[index][id2].type = 0;
1720:
1721:     avg length = CANOBject[index][id1].avg /
1722:              CANOBject[index][id1].length;
1723:     dev length = (cmax - cmin) / CANOBject[index][id1].length;
1724:     if (flag)
1725:        if ( avg length < 10.0 && dev length < 1.0 &&
1726:            CANOBject[index][id1].leaf == 0 )
1727:        {
1728:           CANOBject[index][id1].leaf = 1;
1729:           LeafOBject[leafindex].xindex = index;
1730:           LeafOBject[leafindex].yindex = id1;
1731:           LeafOBject[leafindex].xindex = -1;
1732:        }
1733: }
1734: void CRunLength::RefineLINK ( int i, int *COB_ID, int cob_no )
1735: {
1736:     int nob no, diff, cavg, navg, ldiff;
1737:     int cindex, nindex, clength, nlength, ynpos;
1738:     int NOB ID[MAX OBJECT], min, pos, angle;
1739:     CVt52App *PApp = (CVt52App *) AfxGetApp ();
1740:
1741:     int xc = i * m_sample + m_xorg;
1742:
1743:     nob no = 0;
1744:     angle = 40;
1745:     for (int j = 0; j < OBCount [i+1] ; j++)
1746:        if ( CANOBject [i+1][j].type == REFINE )
1747:           NOB ID [nob no++] = j;
1748:
1749:     if (nob no == 0) return;
1750:     for ( j = 0 ; j < cob_no ; j++ )
1751:     {
1752:        min = ldiff = 0x0ff0;
1753:        cindex = *(COB ID + j);
1754:        cavg = CANOBject[i][cindex].redge + CANOBject[i][cindex].fedge;
1755:        clength = cavg - (CANOBject[i][cindex].fedge<< 1);
1756:        for ( int k = 0 ; k < nob_no; k++ )
1757:        {
1758:           nindex = NOB ID [k];
1759:           navg = CANOBject[i+1][nindex].redge + CANOBject[i+1][nindex].fedge;
1760:           nlength = navg - (CANOBject[i+1][nindex].fedge << 1);
1761:           diff = navg - cavg;
1762:           if (diff < 0) diff = -diff;
1763:
1764:           // add decision rule
1765:           if (min > diff)
1766:           {
1767:              min = diff;
1768:              pos = nindex;
1769:              ynpos = navg;
1770:              ldiff = abs (nlength - clength);
1771:           }
1772:        }
1773:        if (  abs(ynpos - cavg) < angle )
1774:        //if ( ldiff <= clength && abs(ynpos - cavg) < angle )
1775:        {
1776:           CANOBject[i+1][pos].type = NORMAL;
1777:           CANOBject[i][cindex].next id = pos;
1778:           CANOBject[i+1][pos].prev id = cindex;
1779:           if (vdebug) CIC draw line ( oh Console[PApp->Frame_Index],
1780:                 (double) (xc), (double) ((cavg>>1)+m_yorg),
1781:                 (double) (xc+m sample),
1782:                 (double) ((ynpos>>1)+m_yorg), CDE_PEN_YELLOW );
1783:        }
1784:     }
1785: }
1786:
1787: void CRunLength::OverlapOBject ()
1788: {
```

```
1789:     int i, j, ret, jbuf;
1790:     int cid, pid, cidbuf;
1791:
1792:     for ( i = 0 ; i < brcount ; i++ )
1793:     {
1794:        j = BRCOOR [i].xid;
1795:        cid = BRCOOR [i].yid;
1796:        if ( CANOBject [j][cid].count > 12 )
1797:        {
1798:           for ( ; j > 0 ; j-- )
1799:           {
1800:              pid = CANOBject [j][cid].prev id;
1801:              if ( pid == -1 ) break;
1802:
1803:              if ( CANOBject [j][cid].leaf == 1 )
1804:              {
1805:                 if ( CANOBject [j-1][pid].leaf == 1 )
1806:                 {
1807:                    jbuf = j;
1808:                    cidbuf = cid;
1809:                    ret = FindOverlap ( &j, &cid );
1810:                    if ( ret == -1 )
1811:                    {
1812:                       if ( CANOBject[jbuf][cidbuf].count >= 10)
1813:                          MarkLeaf ( jbuf, cidbuf );
1814:                       break;
1815:                    }
1816:                    else if ( ret == 0 && CANOBject[jbuf][cidbuf].count >= 10 )
1817:                       MarkLeaf ( jbuf, cidbuf );
1818:                    pid = CANOBject [j][cid].prev id;
1819:                 }
1820:              }
1821:
1822:              cid = pid;
1823:           }
1824:        }
1825:     }
1826: }
1827:
1828: void CRunLength::MarkLeaf ( int index, int cid )
1829: {
1830:     int i, j, cross;
1831:     int fedge, redge, temp1, temp2;
1832:     CVt52App *PApp = (CVt52App *) AfxGetApp ();
1833:
1834:     j = cid;
1835:     for ( i = index ; i >= 0 ; i-- )
1836:     {
1837:        if ( CANOBject [i][j].leaf == 0 ) return;
1838:        else
1839:        {
1840:           cross = CANOBject [i][j].cross;
1841:           if ( cross == 0 )
1842:           {
1843:              CANOBject [i][j].cross = OBCROSS;
1844:              if (vdebug)
1845:              {
1846:                 temp1 = i * m sample + m xorg;
1847:                 fedge = CANOBject [i][j].fedge;
1848:                 redge = CANOBject [i][j].redge;
1849:                 temp2 = (( fedge + redge ) >> 1) + m yorg;
1850:                 CIC draw cross ( oh Console [PApp->Frame Index],
1851:                    (double) temp1,   (double) temp2,
1852:                    5, 0, CDE PEN YELLOW );
1853:              }
1854:           }
1855:        }
1856:        j = CANOBject [i][j].prev id;
1857:        if ( j == -1 ) return;
1858:     }
1859:
1860: }
1861: int  CRunLength::FindOverlap ( int *xc, int *cid )
1862: {
1863:     int init, end, iid, eid;
```

```
1864:     int i, j, pid;
1865:     int fedge, redge;
1866:     int RNODE, LNODE;
1867:     int plength, clength, nlength;
1868:     CVt52App *PApp = (CVt52App *) AfxGetApp ();
1869:
1870:     i = end = *xc;
1871:     j = eid = *cid;
1872:
1873:     for ( ;; i-- )
1874:     {
1875:        if ( CANOBject [i][j].leaf == 0 ) break;
1876:        else
1877:           CANOBject [i][j].leaf += LEAFDONE;
1878:
1879:        pid = CANOBject [i][j].prev id;
1880:        if ( pid == -1 ) return -1;
1881:        iid = j;
1882:        j = pid;
1883:     }
1884:
1885:     if ( end == i )
1886:     {
1887:        *xc = i;
1888:        *cid = j;
1889:        return 0;
1890:     }
1891:
1892:     init = i+1;
1893:
1894:     i = end+1;
1895:     j = CANOBject [end][eid].next id;
1896:     clength = CANOBject [end][eid].length;
1897:     fedge = CANOBject [end][eid].fedge;
1898:     redge = CANOBject [end][eid].redge;
1899:
1900:      for ( ;; i++ )
1901:     {
1902:        if ( j == -1 )
1903:        {
1904:           RNODE = 0;
1905:           break;
1906:        }
1907:        nlength = CANOBject[i][j].length;
1908:        if ( (clength - nlength) >= nlength )
1909:        {
1910:           if ( fedge <= CANOBject [i][j].fedge &&
1911:                redge >= CANOBject [i][j].redge )
1912:           {
1913:              end = i;
1914:              eid = j;
1915:              RNODE = 1;
1916:           }
1917:           else
1918:              RNODE = 0;
1919:           break;
1920:        }
1921:        j = CANOBject[i][j].next id;
1922:     }
1923:
1924:     i = init-1;
1925:     j = CANOBject [init][iid].prev id;
1926:     clength = CANOBject [init][iid].length;
1927:     fedge = CANOBject [init][iid].fedge;
1928:     redge = CANOBject [init][iid].redge;
1929:
1930:      for ( ;; i-- )
1931:     {
1932:        if ( j == -1 ) return -1;
1933:        plength = CANOBject[i][j].length;
1934:        if ( (clength - plength) >= plength )
1935:        {
1936:           if ( fedge <= CANOBject [i][j].fedge &&
1937:                redge >= CANOBject [i][j].redge )
1938:           {
```

```
1939:              init = i;
1940:              iid = j;
1941:              LNODE = 1;
1942:           }
1943:           else
1944:              LNODE = 0;
1945:           break;
1946:        }
1947:        j = CANOBject[i][j].prev id;
1948:     }
1949:
1950:     if ( LNODE && RNODE )
1951:     {
1952:        int ret, peid;
1953:
1954:        peid = eid;
1955:        ret = CheckLink ( init, &iid, end, &eid );
1956:        if ( ret == -1 ) return -1;
1957:        // two branches was overlapped
1958:        else if ( ret == 1 ) SplitLeaf ( init, iid, end, eid, peid );
1959:        // branch was hidden by leaf
1960:        else if ( ret == 0) ReconnectLink ( init, iid, end, eid );
1961:        if ( vdebug )
1962:        {
1963:           CIC draw line ( oh Console [PApp->Frame Index],
1964:              (double) (init*m sample+m_xorg), (double)
      (CANOBject[init][iid].fedge+m yorg),
1965:              (double) (end*m sample+m_xorg), (double)
      (CANOBject[end][eid].fedge+m yorg),
1966:              CDE PEN YELLOW );
1967:           CIC draw line ( oh Console [PApp->Frame Index],
1968:              (double) (init*m sample+m_xorg), (double)
      (CANOBject[init][iid].redge+m yorg),
1969:              (double) (end*m sample+m_xorg), (double)
      (CANOBject[end][eid].redge+m yorg),
1970:              CDE PEN YELLOW );
1971:        }
1972:     }
1973:
1974:     *xc = init;
1975:     *cid = iid;
1976:
1977:     if ( LNODE == 0 || RNODE == 0 ) return 0;
1978:     else
1979:        return 1;
1980: }
1981:
1982: int  CRunLength::CheckLink ( int init, int *iid, int end, int *eid )
1983: {
1984:     int cob no, nob no, diff, cavg, navg;
1985:     int cid, nid, mcid, mnid, min, j, cross;
1986:     int COB_ID[MAX_OBJECT], NOB_ID[MAX_OBJECT];
1987:
1988:     cob no = 0;
1989:     for ( j = 0; j < OBCount[init] ; j++ )
1990:        if ( CANOBject [init][j].type == NORMAL )
1991:           COB ID [cob no++] = j;
1992:
1993:     nob no = 0;
1994:     for ( j = 0; j < OBCount[end] ; j++ )
1995:        if ( CANOBject [end][j].type == NORMAL )
1996:           NOB ID [nob no++] = j;
1997:
1998:     if (cob_no == 0 || nob_no == 0) return -1;
1999:
2000:     min = 0x0ff0;
2001:     for ( j = 0 ; j < cob_no ; j++ )
2002:     {
2003:        cid = COB ID [j];
2004:        cavg = CANOBject[init][cid].redge + CANOBject[init][cid].fedge;
2005:        for ( int k = 0 ; k < nob_no; k++ )
2006:        {
2007:           nid = NOB ID [k];
2008:           navg = CANOBject[end][nid].redge + CANOBject[end][nid].fedge;
2009:           diff = navg - cavg;
```

```
2010:          if (diff < 0) diff = -diff;
2011:
2012:          // add decision rule
2013:          if (min > diff)
2014:          {
2015:             min = diff;
2016:             mcid = cid;
2017:             mnid = nid;
2018:          }
2019:       }
2020:    }
2021:
2022:    if ( mcid != *iid )        *iid = mcid;
2023:    else if  ( mnid != *eid ) *eid = mnid;
2024:    cid = *iid;
2025:    for ( int i = init ; i <= end ; i++ )
2026:    {
2027:       cross = CANOBject [i][cid].cross;
2028:       if ( cross == OBPARENT || cross == OBCHILD )
2029:          return 1;
2030:       cid = CANOBject [i][cid].next id;
2031:    }
2032:
2033:    return 0;
2034: }
2035:
2036: void CRunLength::SplitLeaf ( int init, int iid, int end, int eid, int peid)
2037: {
2038:    int i, j, cid, found, fdiff, rdiff;
2039:    int parent, child, temp1, temp2, cross;
2040:    int f1, f2, xdist, efedge, eredge, ecent;
2041:    int cfedge, credge, ppid, cpid, ret, length;
2042:    CVt52App *PApp = (CVt52App *) AfxGetApp ();
2043:
2044:
2045:    // Search the Split Object
2046:    cid = iid;
2047:    found = 0;
2048:    for ( i = init ; i <= end ; i++ )
2049:    {
2050:       if ( cid == -1 ) return;
2051:       cross = CANOBject [i][cid].cross;
2052:       if ( CANOBject [i][cid].leaf )
2053:       {
2054:          if ( cross == 0 )
2055:             CANOBject [i][cid].cross = OBCROSS;
2056:          temp1 = CANOBject [i][cid].fedge + CANOBject [i][cid].redge;
2057:          if ( vdebug )
2058:          {
2059:             CIC draw cross ( oh Console [PApp->Frame Index],
2060:                (double) (i*m sample+m xorg), (double) ((temp1>>1)+m_yorg),
2061:                5, 0, CDE PEN YELLOW );
2062:          }
2063:       }
2064:       if ( cross == OBPARENT && found == 0 )
2065:       {
2066:          child = -1;
2067:          for ( j = 0 ; j < OBCount [i] ; j++ )
2068:             if ( CANOBject [i][j].cross == OBCHILD )
2069:                child = j;
2070:          if ( child == -1 ) return;
2071:          found = 1;
2072:       }
2073:
2074:       if ( cross == 0 && found == 1 )
2075:          break;
2076:       if ( found )
2077:       {
2078:          ppid = cid;
2079:          cpid = child;
2080:       }
2081:       cid = CANOBject [i][cid].next id;
2082:       if ( found )
2083:       {
2084:          parent = cid;
```

```
2085:            child = CANOBject [i][child].next id;
2086:        }
2087:    }
2088:
2089:    if ( found == 0 ) return;
2090:
2091:    f1 = CANOBject[init][iid].fedge;
2092:    f2 = CANOBject[init][iid].redge;
2093:    xdist = end - init;
2094:    fdiff = CANOBject[end][eid].fedge - CANOBject[init][iid].fedge;
2095:    rdiff = CANOBject[end][eid].redge - CANOBject[init][iid].redge;
2096:
2097:    for ( ; i <= end ; i++ )
2098:    {
2099:       if ( cid  == -1 ) return;
2100:       if ( i == end  && cid == eid )
2101:       {
2102:          cid = eid;
2103:          ret = peid;
2104:          CANOBject [i-1][ppid].next id = cid;
2105:          CANOBject [i-1][cpid].next id = ret;
2106:          CANOBject [i][cid].prev id = ppid;
2107:          CANOBject [i][ret].prev id = cpid;
2108:          CANOBject [i][cid].count = CANOBject [i-1][ppid].count + 1;
2109:          CANOBject [i][ret].count = CANOBject [i-1][cpid].count + 1;
2110:          cid = CANOBject[i][cid].next id;
2111:          ret = CANOBject[i][ret].next id;
2112:          i++;
2113:          break;
2114:       }
2115:
2116:       efedge = f1 + fdiff * ( i - init ) / xdist;
2117:       eredge = f2 + rdiff * ( i - init ) / xdist;
2118:       ecent  = ( efedge + eredge ) >> 1;
2119:
2120:       cfedge = CANOBject [i][cid].fedge;
2121:       credge = CANOBject [i][cid].redge;
2122:       length = CANOBject [i][cid].length;
2123:       if ( cfedge <= efedge && credge >= eredge )
2124:       {
2125:          temp1 = abs ( cfedge - ecent );
2126:          temp2 = abs ( credge - ecent );
2127:
2128:          if ( temp1 > temp2 )
2129:              ret = OBjectSplit ( i, cid, ecent-temp2, 2 );
2130:          else ret = OBjectSplit ( i, cid, ecent+temp1, 3 );
2131:       }
2132:       else
2133:       {
2134:          found = 0;
2135:          for ( j = 0 ; j < OBCount [i] ; j++ )
2136:          {
2137:             cfedge = CANOBject [i][j].fedge;
2138:             credge = CANOBject [i][j].redge;
2139:             if ( cfedge <= efedge && credge >= eredge )
2140:             {
2141:                ret = cid;
2142:                cid = j;
2143:                found = 1;
2144:             }
2145:          }
2146:          if ( found == 0 )
2147:          {
2148:             temp1 = cid;
2149:             ObjectInit ( i, 0, 0, efedge, eredge, 0, -1, 1 );
2150:             cid = OBCount[i] - 1;
2151:             ret = temp1;
2152:          }
2153:       }
2154:       if ( CANOBject [i][cid].leaf )
2155:       {
2156:          cross = CANOBject [i][cid].cross;
2157:          if ( cross == 0 )
2158:          {
2159:             CANOBject [i][cid].cross = OBCROSS;
```

```
2160:              CANObject [i][ret].cross = OBCROSS;
2161:           }
2162:           found = i * m_sample + m_xorg;
2163:           temp1 = CANObject [i][cid].fedge + CANObject [i][cid].redge;
2164:           temp2 = CANObject [i][ret].fedge + CANObject [i][ret].redge;
2165:           if ( vdebug )
2166:           {
2167:              CIC_draw_cross ( oh_Console [PApp->Frame_Index],
2168:                 (double) found,    (double) ((temp1>>1)+m_yorg),
2169:                 5, 0, CDE_PEN_YELLOW );
2170:              CIC_draw_cross ( oh_Console [PApp->Frame_Index],
2171:                 (double) found,    (double) ((temp2>>1)+m_yorg),
2172:                 5, 0, CDE_PEN_YELLOW );
2173:           }
2174:        }
2175:        CANObject [i-1][ppid].next_id = cid;
2176:        CANObject [i-1][cpid].next_id = ret;
2177:        CANObject [i][cid].prev_id = ppid;
2178:        CANObject [i][ret].prev_id = cpid;
2179:        CANObject [i][cid].count = CANObject [i-1][ppid].count + 1;
2180:        CANObject [i][ret].count = CANObject [i-1][cpid].count + 1;
2181:        ppid = cid;
2182:        cpid = ret;
2183:        cid = CANObject [i][cid].next_id;
2184:        if ( i == end ) ret = CANObject [i][ret].next_id;
2185:     }
2186:
2187:     // Relink Objects
2188:     // Reconnect Parent & Child Object
2189:     Recount ( i, cid, ret );
2190: }
2191:
2192: void CRunLength::ReconnectLink ( int init, int iid, int end, int eid )
2193: {
2194:     int i, first, ppid, cpid, ret;
2195:     int f1, f2, fdiff, rdiff, xdist;
2196:     int efedge, eredge, cid;
2197:
2198:     ppid = iid;
2199:     cpid = -1;
2200:     ret = CANObject [init][iid].next_id;
2201:     f1 = CANObject[init][iid].fedge;
2202:     f2 = CANObject[init][iid].redge;
2203:     xdist = end - init;
2204:     fdiff = CANObject[end][eid].fedge - CANObject[init][iid].fedge;
2205:     rdiff = CANObject[end][eid].redge - CANObject[init][iid].redge;
2206:     first = 1;
2207:
2208:     i = init + 1;
2209:     for ( ; i <= end ; i++ )
2210:     {
2211:        if ( ret == -1 ) return;
2212:        efedge = f1 + fdiff * ( i - init ) / xdist;
2213:        eredge = f2 + rdiff * ( i - init ) / xdist;
2214:
2215:        if ( i == end )
2216:           cid = eid;
2217:        else
2218:        {
2219:           ObjectInit ( i, 0, 0, efedge, eredge, 0, -1, 1 );
2220:           cid = OBCount[i] - 1;
2221:        }
2222:
2223:        CANObject [i-1][ppid].next_id = cid;
2224:        CANObject [i][cid].prev_id = ppid;
2225:        CANObject [i][cid].count = CANObject [i-1][ppid].count + 1;
2226:        if ( first == 0 )
2227:        {
2228:           CANObject [i-1][cpid].next_id = ret;
2229:           CANObject [i][ret].prev_id = cpid;
2230:           CANObject [i][ret].count = CANObject [i-1][cpid].count+1;
2231:        }
2232:        else
2233:        {
2234:           CANObject [i][ret].prev_id = -1;
```

```
2235:            CANObject [i][ret].count = 0;
2236:         }
2237:
2238:         ppid = cid;
2239:         cpid = ret;
2240:         if ( first ) first = 0;
2241:
2242:         ret = CANObject [i][ret].next id;
2243:         if ( i == end )
2244:            cid = CANObject [i][cid].next id;
2245:      }
2246:
2247:
2248:      // Relink Objects
2249:      // Reconnect Parent & Child Object
2250:      Recount ( i, cid, ret );
2251: }
2252:
2253: void CRunLength::Recount ( int i, int pid, int cid )
2254: {
2255:      int ppid, cpid;
2256:      int stop, pcount, ccount;
2257:
2258:      ppid = CANObject [i][pid].prev id;
2259:      cpid = CANObject [i][cid].prev id;
2260:      pcount = CANObject [i-1][ppid].count+1;
2261:      ccount = CANObject [i-1][cpid].count+1;
2262:      stop = 0;
2263:      for ( ; i < xcount ; i++ )
2264:      {
2265:         CANObject [i][pid].count = pcount++;
2266:         if ( pid != cid && stop == 0 )
2267:            CANObject [i][cid].count = ccount++;
2268:         else if ( pid == cid && stop == 0 )
2269:            stop = 1;
2270:
2271:         pid = CANObject [i][pid].next id;
2272:         if ( pid == -1 ) break;
2273:         if ( stop == 0)
2274:         {
2275:            cid = CANObject [i][cid].next id;
2276:            if ( cid == -1 ) stop = 1;
2277:         }
2278:      }
2279: }
2280: void CRunLength::OnRunlength4()
2281: {
2282:      // TODO: Add your control notification handler code here
2283:      ObjectFill ();
2284: }
2285:
2286: void CRunLength::FindLeaves()
2287: {
2288:
2289:      short next id;
2290:      int      x00, y00, x01, y01;
2291:      int    x10, y10, x11, y11;
2292:      int      corect[30][10];
2293:      RECT  CANROI;
2294:      CVt52App *PApp = (CVt52App *) AfxGetApp ();
2295:
2296:      leafcount = 0;
2297:      leafindex = 0;
2298:      rcount = 0;
2299:      for ( int i = 0; i < xcount;i++ )
2300:      {
2301:         for ( int j = 0 ; j < OBCount [i] ; j++ )
2302:         {
2303:            if ( CANObject [i][j].type == NORMAL )
2304:            {
2305:               next id = CANObject[i][j].next id;
2306:               if ( next id != -1  &&
2307:                   (next id & LEAVESMASK) == 0 )
2308:               {
2309:                  FindCandidate ( i, j );
```

```
2310:                }
2311:            }
2312:        }
2313:    }
2314:
2315:    int xref14, yref14, xref34, yref34;
2316:    int x0_size, x1_size, y0_size, y1_size;
2317:    int count, color, sum;
2318:
2319:    for ( i = 0 ; i < rcount ; i++ )
2320:    {
2321:       x00 = LeafRect[i].left;
2322:       x01 = LeafRect[i].right;
2323:       y00 = LeafRect[i].top;
2324:       y01 = LeafRect[i].bottom;
2325:       x0_size = x01 - x00;
2326:       count = 0;
2327:       sum = 0;
2328:
2329:       for ( int j = 0 ; j < rcount ; j++ )
2330:       {
2331:          if ( i != j )
2332:          {
2333:             x10 = LeafRect[j].left;
2334:             x11 = LeafRect[j].right;
2335:             y10 = LeafRect[j].top;
2336:             y11 = LeafRect[j].bottom;
2337:             x1_size = x11 - x10;
2338:             if ( x1_size < x0_size )
2339:             {
2340:                xref14 = x10 + (( x11 - x10 ) >> 2);
2341:                xref34 = x11 - (( x11 - x10 ) >> 2);
2342:                yref14 = y10 + (( y11 - y10 ) >> 2);
2343:                yref34 = y11 - (( y11 - y10 ) >> 2);
2344:                if ( x00 <= xref14 && x01 >= xref34 &&
2345:                     y00 <= yref14 && y01 >= yref34 )
2346:                {
2347:                     corect[i][count+2] = j;
2348:                     sum += i + j;
2349:                     count++;
2350:                }
2351:             }
2352:             else if ( x1_size > x0_size )
2353:             {
2354:                xref14 = x00 + (( x01 - x00 ) >> 2);
2355:                xref34 = x01 - (( x01 - x00 ) >> 2);
2356:                yref14 = y00 + (( y01 - y00 ) >> 2);
2357:                yref34 = y01 - (( y01 - y00 ) >> 2);
2358:                if ( x10 <= xref14 && x11 >= xref34 &&
2359:                     y10 <= yref14 && y11 >= yref34 )
2360:                {
2361:                     corect[i][count+2] = j;
2362:                     sum += i + j;
2363:                     count++;
2364:                }
2365:             }
2366:             else
2367:             {
2368:                y0_size = y01 - y00;
2369:                y1_size = y11 - y10;
2370:                if (y1_size < y0_size)
2371:                {
2372:                   xref14 = x10 + (( x11 - x10 ) >> 2);
2373:                   xref34 = x11 - (( x11 - x10 ) >> 2);
2374:                   yref14 = y10 + (( y11 - y10 ) >> 2);
2375:                   yref34 = y11 - (( y11 - y10 ) >> 2);
2376:                   if ( x00 <= xref14 && x01 >= xref34 &&
2377:                        y00 <= yref14 && y01 >= yref34 )
2378:                   {
2379:                        corect[i][count+2] = j;
2380:                      sum += i + j;
2381:                      count++;
2382:                   }
2383:                }
2384:                else
```

```
2385:                {
2386:                   xref14 = x00 + (( x01 - x00 ) >> 2);
2387:                   xref34 = x01 - (( x01 - x00 ) >> 2);
2388:                   yref14 = y00 + (( y01 - y00 ) >> 2);
2389:                   yref34 = y01 - (( y01 - y00 ) >> 2);
2390:                   if ( x10 <= xref14 && x11 >= xref34 &&
2391:                        y10 <= yref14 && y11 >= yref34 )
2392:                   {
2393:                       corect[i][count+2] = j;
2394:                       sum += i + j;
2395:                       count++;
2396:                   }
2397:                }
2398:             }
2399:          }
2400:       }
2401:
2402:       corect[i][0] = count;
2403:       corect[i][1] = sum;
2404:       if (vdebug)
2405:       {
2406:          if ( count ) color = CDE PEN WHITE;
2407:          else       color = CDE PEN YELLOW;
2408:          CIC draw rect ( oh Console[PApp->Frame Index],
2409:             (double) x00, (double) y00,   (double) ( x01 - x00 ),
2410:             (double) ( y01 - y00 ), color);
2411:       }
2412:    }
2413:
2414:    for ( i = 0; i < rcount; i++)
2415:    {
2416:       count = corect[i][0];
2417:       if (count == 0)
2418:          leafinit[i] |= LEAFINITFLAG;
2419:
2420:       if (count == 1)
2421:       {
2422:          sum = corect[i][1];
2423:          for ( int j = 0; j < rcount; j++)
2424:          {
2425:             if (i != j)
2426:             {
2427:                if (sum == corect[j][1])
2428:                {
2429:                   if (i > j)
2430:                   {
2431:                      next id = j;
2432:                      rectcount[i] = next id;
2433:                      corect[i][0] |= OVERLAP WINDOW;
2434:                   }
2435:                   else
2436:                   {
2437:                      next id = i;
2438:                      rectcount[j] = next id;
2439:                      corect[j][0] |= OVERLAP WINDOW;
2440:                   }
2441:
2442:                   ROIMerge ( &LeafRect[i], &LeafRect[j], &CANROI, i, j );
2443:
2444:                   LeafRect[next id].left = CANROI.left;
2445:                   LeafRect[next id].top  = CANROI.top;
2446:                   LeafRect[next id].right = CANROI.right;
2447:                   LeafRect[next id].bottom = CANROI.bottom;
2448:                   leafinit[next id] |= LEAFINITFLAG;
2449:
2450:                   if (vdebug)
2451:                      CIC draw rect ( oh Console[PApp->Frame Index],
2452:                         (double) CANROI.left, (double) CANROI.top,
2453:                         (double) (CANROI.right - CANROI.left),
2454:                         (double) (CANROI.bottom - CANROI.top),
2455:                         CDE PEN RED);
2456:                }
2457:             }
2458:          }
2459:       }
```

```
2460:        else if ((count & 0xff) > 1 && (count & OVERLAP_WINDOW) == 0)
2461:        {
2462:           sum = corect[i][1];
2463:           xref14 = 0;
2464:           int min = 0xff0;
2465:           for ( int j = 0; j < count; j++)
2466:           {
2467:              xref34 = corect[i][j+2];
2468:              xref14 += corect[xref34][1];
2469:              if ( min > xref34 )
2470:                 min = xref34;
2471:           }
2472:
2473:           if ( sum == xref14)
2474:           {
2475:              CANROI.left = LeafRect[i].left;
2476:              CANROI.right = LeafRect[i].right;
2477:              CANROI.top = LeafRect[i].top;
2478:              CANROI.bottom = LeafRect[i].bottom;
2479:              for ( j = 0; j < count; j++)
2480:              {
2481:                 xref14 = corect[i][j+2];
2482:                 if ( xref14 != min )
2483:                       corect[xref14][0] |= OVERLAP WINDOW;
2484:                 ROIMerge ( &CANROI, &LeafRect[xref14], &CANROI, i, xref14 );
2485:              }
2486:
2487:              if ( i > min )
2488:              {
2489:                 next id = min;
2490:                 rectcount[min] = next id;
2491:                 corect[i][0] |= OVERLAP WINDOW;
2492:              }
2493:              else
2494:              {
2495:                 next id = i;
2496:                 rectcount[i] = next id;
2497:                 corect[min][0] |= OVERLAP WINDOW;
2498:              }
2499:              LeafRect[next id].left = CANROI.left;
2500:              LeafRect[next id].top  = CANROI.top;
2501:              LeafRect[next id].right = CANROI.right;
2502:              LeafRect[next id].bottom = CANROI.bottom;
2503:              leafinit[next id] |= LEAFINITFLAG;
2504:
2505:              if (vdebug)
2506:                 CIC draw rect ( oh Console[PApp->Frame Index],
2507:                       (double) CANROI.left, (double) CANROI.top,
2508:                       (double) (CANROI.right - CANROI.left),
2509:                       (double) (CANROI.bottom - CANROI.top),
2510:                       CDE PEN RED);
2511:           }
2512:        }
2513:
2514:     }
2515:
2516:     MainBranch ();
2517:     for ( i = 0; i < rcount; i++)
2518:     {
2519:        count = corect[i][0];
2520:        if ( (count & OVERLAP_WINDOW) == 0 )
2521:        {
2522:           XRunRect.x = LeafRect[i].left;
2523:           XRunRect.y = LeafRect[i].top;
2524:           XRunRect.w = (LeafRect[i].right  - LeafRect[i].left);
2525:           XRunRect.h = (LeafRect[i].bottom - LeafRect[i].top);
2526:           leafindex = rectcount[i];
2527:           // to use leaf ocject count
2528:           XScan ( 0 );
2529:           MakeLeaf ();
2530:        }
2531:     }
2532:     if (vdebug) CIC_force_refresh  ( oh_Console[PApp->Frame_Index] );
2533:
2534: }
```

```
2535: void CRunLength::ROIMerge ( RECT *rect1, RECT *rect2, RECT *mroi,
2536:                             int sr1, int sr2)
2537: {
2538:     int x00, y00, x01, y01;
2539:     int x10, y10, x11, y11;
2540:     int i, j, index, frid, toid;
2541:     short ob id;
2542:     CVt52App *PApp = (CVt52App *) AfxGetApp ();
2543:
2544:     x00 = rect1->left;
2545:     y00 = rect1->top;
2546:     x01 = rect1->right;
2547:     y01 = rect1->bottom;
2548:
2549:     x10 = rect2->left;
2550:     y10 = rect2->top;
2551:     x11 = rect2->right;
2552:     y11 = rect2->bottom;
2553:
2554:     if (x00 >= x10)
2555:         mroi->left = x10;
2556:     else mroi->left = x00;
2557:
2558:     if (x01 <= x11)
2559:         mroi->right = x11;
2560:     else mroi->right = x01;
2561:
2562:     if (y00 >= y10)
2563:         mroi->top = y10;
2564:     else mroi->top = y00;
2565:
2566:     if (y01 <= y11)
2567:         mroi->bottom = y11;
2568:     else mroi->bottom = y01;
2569:
2570:     i = 0;
2571:     if ( leafinit[sr1] <= leafinit[sr2] )
2572:     {
2573:        frid = sr1;
2574:        toid = sr2;
2575:     }
2576:     else
2577:     {
2578:        frid = sr2;
2579:        toid = sr1;
2580:     }
2581:
2582:     if ( leafinit[frid] <= leafinit[toid] )
2583:     {
2584:        if (leafend[frid] == leafinit[toid])
2585:        {
2586:           i = leafend[frid] - leafinit[frid];
2587:           index = leafend[frid] - 1;
2588:        }
2589:        else if ( leafend[frid] == (leafinit[toid] - 1) )
2590:        {
2591:           i = leafend[frid] - leafinit[frid] + 1;
2592:           index = leafend[frid];
2593:        }
2594:     }
2595:
2596:     x00 = Leaf[frid].ob id[i-1] & 0xff;
2597:     x11 = Leaf[toid].ob id[0] & 0xff;
2598:     if ( i )
2599:     {
2600:        for ( j = 0 ;; j++ )
2601:        {
2602:           ob id = Leaf[toid].ob id[j];
2603:           if (ob id == -1) break;
2604:           if (j == 0)
2605:           {
2606:              CANOBject[index][x00].next id = ob id;
2607:              if (vdebug)
2608:              {
2609:                 x10 = index*m sample + m xorg;
```

```
2610:                 y00 = ( CANObject[index][x00].fedge +
2611:                         CANObject[index][x00].redge ) >> 1;
2612:                 y11 = ( CANObject[index+1][x11].fedge +
2613:                         CANObject[index+1][x11].redge ) >> 1;
2614:                 CIC draw line ( oh Console[PApp->Frame Index],
2615:                    (double) x10, (double) (y00+m_yorg),
2616:                    (double) (x10+m sample),
2617:                    (double) (y11+m_yorg),  CDE_PEN_YELLOW );
2618:              }
2619:           }
2620:           Leaf[frid].ob id[i++] = ob id;
2621:
2622:        }
2623:        Leaf[frid].ob id[i] = -1;
2624:     }
2625: }
2626:
2627: void CRunLength::FindCandidate ( int i, int j )
2628: {
2629:    int linit[10], lend[10], lmin[10], lmax[10];
2630:    int index, next id, count;
2631:    int fedge, redge, gapsize;
2632:    int leafsearch, xc, chain;
2633:
2634:    CVt52App *PApp = (CVt52App *) AfxGetApp ();
2635:
2636:    index = i;
2637:    count = 0;
2638:    xc = index * m sample;
2639:    leafsearch = 1;
2640:    lmin [count] = 0x0ff0;
2641:    lmax [count] = 0;
2642:    for ( int k = 0 ; k < 10 ; k++ )
2643:    {
2644:       linit[count] = 0;
2645:       lend [count] = 0;
2646:    }
2647:
2648:    chain = 0;
2649:    leafinit[leafcount] = index;
2650:    for ( ; index < xcount ; index++, xc += m_sample )
2651:    {
2652:       next id = CANObject [index][j].next id;
2653:       CANObject[index][j].next id |= LEAVESMASK;
2654:
2655:       if ( (next_id & LEAVESMASK) == 0 )
2656:       {
2657:          if (leafsearch == 1)
2658:          {
2659:             if (CANObject[index][j].leaf == 1)
2660:             {
2661:                fedge = CANObject [index][j].fedge;
2662:                redge = CANObject [index][j].redge;
2663:                linit [count] = index;
2664:                leafsearch = 2;
2665:                gapsize = 0;
2666:                lend[count] = 0;
2667:
2668:                if ( lmin[count] > fedge ) lmin[count] = fedge;
2669:                if ( lmax[count] < redge ) lmax[count] = redge;
2670:             }
2671:          }
2672:          else if (leafsearch == 2)
2673:          {
2674:             if (CANObject[index][j].leaf == 1)
2675:             {
2676:                fedge = CANObject [index][j].fedge;
2677:                redge = CANObject [index][j].redge;
2678:
2679:                if ( lmin[count] > fedge ) lmin[count] = fedge;
2680:                if ( lmax[count] < redge ) lmax[count] = redge;
2681:
2682:                gapsize = 0;
2683:                lend [count] = index;
2684:             }
```

```
2685:              else
2686:              {
2687:                 if (CANOBject[index][j].type == NORMAL)
2688:                 {
2689:                    fedge = CANOBject [index][j].fedge;
2690:                    redge = CANOBject [index][j].redge;
2691:
2692:                    if ( lmin[count] > fedge ) lmin[count] = fedge;
2693:                    if ( lmax[count] < redge ) lmax[count] = redge;
2694:                 }
2695:
2696:                 gapsize++;
2697:                 if (gapsize > 4)
2698:                 {
2699:                    lend[count] = linit[count];
2700:                    count++;
2701:                    gapsize = 0;
2702:                    leafsearch = 1;
2703:                    lmin[count] = 0x0ff0;
2704:                    lmax[count] = 0;
2705:                 }
2706:              }
2707:           }
2708:
2709:           if ( next_id & NEXT_EMPTY_FLAG )
2710:           {
2711:              index++;
2712:              xc += m sample;
2713:              Leaf[leafcount].ob id [chain++] = j | NEXT EMPTY FLAG;
2714:           }
2715:           else
2716:              Leaf[leafcount].ob id [chain++] = j;
2717:        }
2718:        else if ( next_id == -1 )
2719:        {
2720:           Leaf[leafcount].ob id [chain++] = j;
2721:           break;
2722:        }
2723:        else
2724:        {
2725:           Leaf[leafcount].ob id [chain++] = j;
2726:           break;
2727:        }
2728:
2729:        j = next id & 0xff;
2730:     }
2731:     if ( ( leafsearch == 2 ) )
2732:     {
2733:        if ( lend [count] == 0 ) lend [count] = linit [count];
2734:        count++;
2735:     }
2736:     leafend[leafcount] = index;
2737:     Leaf[leafcount].ob id[chain] = -1;
2738:
2739:
2740:     int x0, y0, xsize, ysize;
2741:     for ( index = 0; index < count; index++ )
2742:     {
2743:
2744:        x0 = linit[index] * m sample - 25 + m xorg;
2745:        y0 = lmin[index] - 5 + m yorg;
2746:        xsize = (lend[index] - linit[index]) * m sample + 50;
2747:        ysize = lmax[index] - lmin[index] + 10;
2748:        /*
2749:        XRunRect.x = x0;
2750:        XRunRect.y = y0;
2751:        XRunRect.w = xsize;
2752:        XRunRect.h = ysize;
2753:        */
2754:
2755:        LeafRect[rcount].left = x0;
2756:        LeafRect[rcount].top  = y0;
2757:        LeafRect[rcount].right  = x0 + xsize;
2758:        LeafRect[rcount].bottom = y0 + ysize;
2759:        rectcount[rcount++] = leafcount;
```

```
2760:     }
2761:     if (count > 0) leafcount++;
2762:
2763: }
2764:
2765: void CRunLength::XScan ( int para )
2766: {
2767:        // TODO: Add your control notification handler code here
2768:     CPB oh oh RunLength;
2769:     c Int32  flag;
2770:     CVt52App *PApp = (CVt52App *) AfxGetApp ();
2771:     c UInt8  prev, next;
2772:     int   diff, color, yc;
2773:     int psign, csign, nsign;
2774:     int pbin, nbin;
2775:
2776:     // allocate the ROI Resource
2777:     oh RunLength = CPB alloc ();
2778:     CPB bind ( oh RunLength, windowHandle[0], 1, &flag);
2779:     CPB set window root ( oh RunLength, XRunRect.x, XRunRect.y,
2780:                     XRunRect.w+1, XRunRect.h+1, 0, 0);
2781:
2782:     yc = 0;
2783:     for ( int j = 0 ; j < XRunRect.h; j += m_sample, yc++ )
2784:     {
2785:        int xc = 0;
2786:        for ( int i = 1; i < XRunRect.w ; i++)
2787:        {
2788:           prev = CPB pixel get (oh RunLength, i-1, j);
2789:           next = CPB pixel get (oh RunLength, i+1, j);
2790:           diff = next - prev;
2791:
2792:           if ( prev >= thres ) pbin = 0xff;
2793:           else                 pbin = 0;
2794:
2795:           if ( next >= thres ) nbin = 0xff;
2796:           else                 nbin = 0;
2797:
2798:           if ( ( abs(diff) > m_diff ) && ( pbin != nbin ) )
2799:           {
2800:              if ( xc >= MAX_OBJECT )
2801:              {
2802:                 int ret;
2803:                 ret = AfxMessageBox ( "X축 Edge 수 초과", MB OKCANCEL );
2804:                 if ( ret == IDCANCEL )
2805:                 {
2806:                    // delete the resource
2807:                    CPB set unbound ( oh RunLength );
2808:                    CPB free (oh RunLength );
2809:                    return;
2810:                 }
2811:              }
2812:
2813:              XEdge[yc][xc].edge = i;
2814:              if ( diff >= 0 )
2815:              {
2816:                 XEdge[yc][xc++].sign = PLUS;
2817:                 color = CDE PEN RED;
2818:              }
2819:              else
2820:              {
2821:                 XEdge[yc][xc++].sign = MINUS;
2822:                 color = CDE PEN BLUE;
2823:              }
2824:           }
2825:        }
2826:         XEdge[yc][xc].edge = -1;
2827:        // Remove the illegal Edges
2828:        int PLUS Count = 0;
2829:        for ( i = 1; i < xc; i++ )
2830:        {
2831:           psign = XEdge[yc][i-1].sign;
2832:           csign = XEdge[yc][i].sign;
2833:           if (psign == MINUS)
2834:           {
```

```
2835:              PLUS Count = 0;
2836:              color = CDE PEN BLUE;
2837:           }
2838:           else
2839:           {
2840:              PLUS Count++;
2841:              color = CDE PEN RED;
2842:           }
2843:
2844:           if (psign == csign)
2845:           {
2846:              int   isolate = 0;
2847:              for ( int k = XEdge[yc][i-1].edge ;
2848:                      k < XEdge[yc][i].edge ; k++ )
2849:              {
2850:                 int curr = CPB pixel get (oh RunLength, k,   j);
2851:                 int next = CPB_pixel_get (oh_RunLength, k+1, j);
2852:
2853:                 if ( (next - curr) > 0 )
2854:                 {
2855:                    if (csign == MINUS) isolate++;
2856:                 }
2857:                 else if ( (next - curr) < 0 )
2858:                 {
2859:                    if ( csign == PLUS ) isolate++;
2860:                 }
2861:                 else;
2862:              }
2863:
2864:              if (isolate == 0)
2865:              {
2866:                 color = CDE PEN YELLOW;
2867:                 if (PLUS_Count == 1) color = CDE_PEN_RED;
2868:              }
2869:              else
2870:              {
2871:                 if (PLUS Count > 1) color = CDE_PEN_YELLOW;
2872:                 PLUS Count = 0;
2873:              }
2874:           }
2875:           else if ( (psign == PLUS) && (csign == MINUS) )
2876:           {
2877:              if (PLUS_Count > 1) color = CDE_PEN_YELLOW;
2878:           }
2879:           // draw the cross hair line (application point)
2880:           if ( color == CDE PEN YELLOW )
2881:              XEdge[yc][i-1].sign = VIRTUAL;
2882:           if (vdebug) CIC draw cross ( oh Console[PApp->Frame Index],
2883:                       (double) (XEdge[yc][i-1].edge+XRunRect.x),
2884:                       (double) (j+XRunRect.y), 3 , 0, color );
2885:        }
2886:        if ( i == xc )
2887:        {
2888:           psign = XEdge[yc][i-1].sign;
2889:           if (psign == MINUS) color = CDE_PEN_BLUE;
2890:           else
2891:              if (PLUS_Count == 0) color = CDE_PEN_RED;
2892:              else
2893:              {
2894:                 color = CDE PEN YELLOW;
2895:                 XEdge[yc][i-1].sign = VIRTUAL;
2896:              }
2897:
2898:           if (vdebug) CIC draw cross ( oh Console[PApp->Frame Index],
2899:                       (double) (XEdge[yc][i-1].edge+XRunRect.x),
2900:                       (double) (j+XRunRect.y), 3 , 0, color );
2901:        }
2902:
2903:        int k = 0;
2904:        short sign;
2905:        for ( i = 0; i < xc; i++ )
2906:        {
2907:           sign = XEdge[yc][i].sign;
2908:           if (sign != VIRTUAL)
2909:           {
```

```
2910:              diff = XEdge[yc][i].edge;
2911:              XEdge[yc][k].edge = diff;
2912:              XEdge[yc][k++].sign = sign;
2913:           }
2914:        }
2915:
2916:        XEdge[yc][k].edge = -1;
2917:     }
2918:
2919:     //  Merge the similar type Object
2920:     int init, end, temp;
2921:     int min, max, avg;
2922:     c UInt8 curr;
2923:     int SKIPFLAG, xinit, xend;
2924:
2925:     max = 0;
2926:     init = 0; end = 0;
2927:     yc = 0;
2928:     for ( j = 0 ; j < XRunRect.h; j += m_sample, yc++ )
2929:     {
2930:        for ( int i = 0 ;; i++ )
2931:        {
2932:           if ( XEdge[yc][i].edge == -1 || XEdge[yc][i+1].edge == -1) break;
2933:
2934:           csign = XEdge[yc][i].sign;
2935:           nsign = XEdge[yc][i+1].sign;
2936:           if ( (csign == MINUS) && (nsign == PLUS) )
2937:           {
2938:              temp = XEdge[yc][i+1].edge - XEdge[yc][i].edge;
2939:              if ( temp <= 5 && para == 1 )
2940:              {
2941:                 if (vdebug)
2942:                    CIC draw line ( oh Console[PApp->Frame Index],
2943:                       (double) (XEdge[yc][i].edge+XRunRect.x),
2944:                       (double) (j+XRunRect.y),
2945:                       (double) (XEdge[yc][i+1].edge+XRunRect.x),
2946:                       (double) (j+XRunRect.y), CDE_PEN_ORANGE );
2947:                 XEdge[yc][i].edge |= BRANCHFLAG;
2948:              }
2949:
2950:              if ( temp > max )
2951:              {
2952:                 max = temp;
2953:                 init = XEdge[yc][i].edge;
2954:                 end  = XEdge[yc][i+1].edge;
2955:              }
2956:           }
2957:        }
2958:     }
2959:
2960:     if ( para )
2961:     {
2962:        // delete the resource
2963:        CPB set unbound ( oh RunLength );
2964:        CPB free (oh RunLength );
2965:        return;
2966:     }
2967:
2968:     yc = 0;
2969:     for ( j = 0 ; j < XRunRect.h; j += m_sample, yc++ )
2970:     {
2971:        for ( int i = 1 ; ; i++ )
2972:        {
2973:           if ( XEdge[yc][i-1].edge == -1 || XEdge[yc][i].edge == -1 ||
2974:                XEdge[yc][i+1].edge == -1 ) break;
2975:           psign = XEdge[yc][i-1].sign & 0xff;
2976:           csign = XEdge[yc][i].sign   & 0xff;
2977:           nsign = XEdge[yc][i+1].sign & 0xff;
2978:
2979:           if ( (psign == MINUS && csign == MINUS && nsign == PLUS) ||
2980:                (psign == MINUS && csign == PLUS  && nsign == PLUS) ||
2981:               (psign == csign && csign == nsign && nsign == MINUS) )
2982:           {
2983:              temp = XEdge[yc][i].edge;
2984:              if ( (temp >= init) && (temp <= end) )
```

```
2985:                     XEdge[yc][i].sign |= EDGE_SKIP_FLAG;
2986:
2987:          }
2988:       }
2989:
2990:       // Create Objects
2991:       SKIPFLAG = 0;
2992:       for ( i = 0 ; ; i++)
2993:       {
2994:          if (XEdge[yc][i].edge == -1 || XEdge[yc][i+1].edge == -1 ) break;
2995:          if (SKIPFLAG == 0)
2996:          {  csign = XEdge[yc][i].sign;
2997:             xinit = XEdge[yc][i].edge;
2998:          }
2999:          nsign = XEdge[yc][i+1].sign;
3000:
3001:          if (nsign & EDGE_SKIP_FLAG)
3002:              SKIPFLAG = 1;
3003:          else SKIPFLAG = 0;
3004:
3005:          if ( csign == MINUS && nsign == PLUS)
3006:          {
3007:             xend = XEdge[yc][i+1].edge;
3008:             if (xinit >= xend)
3009:                temp  = 0;
3010:             min = 256; max = 0; avg = 0;
3011:             for (int k = init ; k <= end ; k++)
3012:             {
3013:                curr = CPB_pixel_get (oh_RunLength, k, j);
3014:                if ( min > curr ) min = curr;
3015:                if ( max < curr ) max = curr;
3016:                avg += curr;
3017:             }
3018:
3019:             color = CDE_PEN_GREEN;
3020:             if (vdebug) CIC_draw_line ( oh_Console[PApp->Frame_Index],
3021:                      (double) (xinit+XRunRect.x),
3022:                      (double) (j+XRunRect.y), (double) (xend+XRunRect.x),
3023:                      (double) (j+XRunRect.y), color );
3024:          }
3025:       }
3026:    }
3027:
3028:    yc = 0; max = 0; avg = 0; xinit = 0;
3029:    for ( j = 0 ; j < XRunRect.h; j += m_sample, yc++ )
3030:    {
3031:       int k = 0;
3032:       short sign;
3033:       for ( int i = 0;; i++ )
3034:       {
3035:          if ( XEdge[yc][i].edge == -1) break;
3036:          sign = XEdge[yc][i].sign;
3037:          if ( (sign & EDGE_SKIP_FLAG) == 0 )
3038:          {
3039:             diff = XEdge[yc][i].edge;
3040:             XEdge[yc][k].edge = diff;
3041:             XEdge[yc][k].sign = sign;
3042:             if (k > 0)
3043:             {
3044:                if ( (XEdge[yc][k-1].sign & 0xff) == MINUS &&
3045:                   (XEdge[yc][k].sign & 0xff) == PLUS )
3046:                {
3047:                   temp = XEdge[yc][k].edge - XEdge[yc][k-1].edge;
3048:                   if ( max < temp )
3049:                   {
3050:                      max = temp;
3051:                      xinit = yc;
3052:                      avg = k-1;
3053:                   }
3054:                }
3055:             }
3056:             k++;
3057:          }
3058:       }
3059:       XEdge[yc][k].edge = -1;
```

```
3060:     }
3061:
3062:     XEdge[xinit][avg].edge |= XEDGE MAXPOS;
3063:     // delete the resource
3064:     CPB set unbound ( oh RunLength );
3065:     CPB free (oh RunLength );
3066:
3067: }
3068:
3069: void CRunLength::MakeLeaf()
3070: {
3071:     int i, j, iend, yinit, yend;
3072:     int index, ob id, xc;
3073:     int xcross[30][10];
3074:     int ycross[30],icount;
3075:     int start, sindex, level;
3076:     int first, xmax, outlet;
3077:     int ymin, ymax, temp1, temp2, count;
3078:     int ret;
3079:     double ratio;
3080:     LINE BRLINE[10];
3081:
3082:     CVt52App *PApp = (CVt52App *) AfxGetApp ();
3083:
3084:     for ( i = 0; i < 30; i ++ )
3085:     {
3086:        ycross[i] = 0;
3087:        for ( j = 0 ; j < 10 ; j++)
3088:           xcross[i][j] = 0;
3089:     }
3090:
3091:     i = ( XRunRect.x - m xorg ) / m sample;
3092:     if ( XRunRect.x % m_sample ) i++;
3093:
3094:     iend = ( XRunRect.x + XRunRect.w ) / m sample;
3095:     xc = leafinit [leafindex] & 0x3ff;
3096:     if ( i > xc )
3097:        index = i - xc;
3098:     else
3099:     {
3100:        i = xc;
3101:        index = 0;
3102:     }
3103:
3104:     start = i;
3105:     sindex = index;
3106:
3107:     xc = i * m sample + m xorg;
3108:
3109:     xmax = 0; ymax = 0;
3110:     for ( icount = 0 ; i <= iend ; i++, index++, icount++, xc += m_sample)
3111:     {
3112:        ob id = Leaf[leafindex].ob id[index];
3113:
3114:        if ( ob_id != -1)
3115:        {
3116:           if ( ob_id & NEXT_EMPTY_FLAG )
3117:           {
3118:              i++;
3119:              xc += m sample;
3120:              ob id &= 0xff;
3121:           }
3122:           if ( CANOBject[i][ob id].type == NORMAL ||
3123:                CANOBject[i][ob id].type == MAINBRANCH )
3124:           {
3125:              yinit = CANOBject[i][ob id].fedge;
3126:              yend  = CANOBject[i][ob id].redge;
3127:
3128:               temp1 = yend - yinit;
3129:              if ( ymax < temp1 )
3130:                 ymax = temp1;
3131:
3132:              Cross Check ( icount, index, yinit, yend, xc,
3133:                        &xmax, &xcross[0][0], &ycross[0]);
3134:           }
```

```
3135:        }
3136:        else if (ob_id == -1)
3137:        {
3138:           break;
3139:        }
3140:     }
3141:
3142:     i = start;
3143:     xc = i * m_sample + m_xorg;
3144:     index = sindex;
3145:     level = (XRunRect.h / m_sample) >> 1;
3146:     int ipos, xpos, curr_id, next_id, LEAF;
3147:     LEAF = 0;
3148:     Leaf[leafindex].LEAF = 0;
3149:     for ( int k = 0 ; k < icount ; k++ )
3150:     {
3151:        if ( ycross[k] >= level )
3152:        {
3153:           LEAF = 1;
3154:           Leaf[leafindex].LEAF = 1;
3155:           break;
3156:        }
3157:     }
3158:
3159:     if (xmax)
3160:        ratio = (double) ymax / (double) xmax;
3161:     else outlet = XYNONE;
3162:
3163:     if ( ratio >= 1.2 )
3164:        outlet = YLARGE;
3165:     else if ( ratio <= 0.8 )
3166:        outlet = XYEQUAL;
3167:     else
3168:        outlet = XLARGE;
3169:
3170:     if ( LEAF )
3171:     {
3172:        first = 1;
3173:        level = 0; count = 0;
3174:        ymin = 0xff0; ymax = 0;
3175:        for ( k = 0 ; k < (icount-1);
3176:                        k++, index++, i++, xc += m_sample )
3177:        {
3178:           curr_id = Leaf[leafindex].ob_id[index];
3179:           next_id = Leaf[leafindex].ob_id[index+1];
3180:           if ( (curr_id & LEAVESFLAG) && (next_id & LEAVESFLAG) )
3181:           {
3182:              count++;
3183:              if ( first )
3184:              {
3185:                 first = 0;
3186:                 count++;
3187:                 Leaf [leafindex].xinit = i;
3188:                 Leaf [leafindex].xend  = i+1;
3189:                 temp1 = CANOBject[i][curr_id&0xff].fedge;
3190:                 if ( ymin > temp1 )
3191:                 {
3192:                    ymin = temp1;
3193:                    Leaf [leafindex].yinit = temp1;
3194:                 }
3195:                 temp2 = CANOBject[i][curr_id&0xff].redge;
3196:                 if ( ymax < temp2 )
3197:                 {
3198:                    ymax = temp2;
3199:                    Leaf [leafindex].yend = temp2;
3200:                 }
3201:                 level += temp1 + temp2;
3202:              }
3203:              else
3204:                 Leaf [leafindex].xend = i+1;
3205:
3206:
3207:              temp1 = CANOBject[i+1][next_id&0xff].fedge;
3208:              if ( ymin > temp1 )
3209:              {
```

```
3210:                ymin = temp1;
3211:                Leaf [leafindex].yinit = temp1;
3212:             }
3213:             temp2 = CANObject[i+1][next_id&0xff].redge;
3214:             if ( ymin > temp2 )
3215:             {
3216:                ymin = temp2;
3217:                Leaf [leafindex].yend = temp2;
3218:             }
3219:
3220:             level += temp1 + temp2;
3221:             ipos = i + 1;
3222:             xpos = xc + m_sample;
3223:             if ( curr_id & NEXT_EMPTY_FLAG )
3224:             {
3225:                ipos++;
3226:                xpos += m_sample;
3227:             }
3228:
3229:             CIC_draw_line ( oh_Console[PApp->Frame_Index],
3230:                (double) (xc),
3231:                (double) (CANObject[i][curr_id&0xff].fedge+m_yorg),
3232:                (double) (xpos),
3233:                (double) (CANObject[ipos][next_id&0xff].fedge+m_yorg),
     CDE_PEN_RED );
3234:
3235:             CIC_draw_line ( oh_Console[PApp->Frame_Index],
3236:                (double) (xc),
3237:                (double) (CANObject[i][curr_id&0xff].redge+m_yorg),
3238:                (double) (xpos),
3239:                (double) (CANObject[ipos][next_id&0xff].redge+m_yorg),
     CDE_PEN_RED );
3240:          }
3241:          if ( curr_id & NEXT_EMPTY_FLAG)
3242:          {
3243:             i++;
3244:             xc += m_sample;
3245:          }
3246:       }
3247:       temp1 = ( Leaf[leafindex].xinit +
3248:                 Leaf[leafindex].xend );
3249:       temp1 = (temp1 * m_sample) >> 1;
3250:       Leaf [leafindex].xcent = temp1 + m_xorg;
3251:       if (count)
3252:       {
3253:          temp2 = level / ( count << 1 );
3254:          Leaf [leafindex].ycent = temp2 + m_yorg;
3255:          CIC_draw_cross ( oh_Console[PApp->Frame_Index],
3256:             (double) (temp1+m_xorg), (double) (temp2+m_yorg),
3257:                5, 0, CDE_PEN_BLUE );
3258:       }
3259:    }
3260:    else
3261:       Leaf[leafindex].ycent = XRunRect.y + (XRunRect.h >>1);
3262:
3263:    ret = SmallBranch ( outlet, sindex, icount, &BRLINE[0], 1 );
3264:
3265:    if ( ret == 0 && leafindex && outlet != XYNONE )
3266:       SearchBranch ( outlet, start, sindex, icount );
3267: }
3268:
3269: void CRunLength::Cross_Check (int icount, int index, int yinit, int yend, int
     xref,
3270:                               int *xmax, int *xcross, int *ycross)
3271: {
3272:    int yc, xinit, xend;
3273:    int csign, nsign, i;
3274:    int maxflag, temp;
3275:
3276:    yc = 0;
3277:    yinit -= (XRunRect.y - m_yorg);
3278:    yend  -= (XRunRect.y - m_yorg);
3279:    for ( int j = 0 ; j < XRunRect.h ; j += m_sample, yc++)
3280:    {
3281:       if ( j >= yinit && j <= yend )
```

```
3282:        {
3283:           for ( i = 0 ; ; i++)
3284:           {
3285:              xinit = XEdge[yc][i].edge;
3286:              if ( xinit & XEDGE_MAXPOS )
3287:              {
3288:                 xinit &= 0xff;
3289:                 maxflag = 1;
3290:              }
3291:              else
3292:                 maxflag = 0;
3293:
3294:              xend  = XEdge[yc][i+1].edge;
3295:              if ( xinit == -1 || xend == -1 ) break;
3296:              csign = XEdge[yc][i].sign;
3297:              nsign = XEdge[yc][i+1].sign;
3298:              xinit += XRunRect.x;
3299:              xend  += XRunRect.x;
3300:              if ( csign == MINUS &&  nsign == PLUS )
3301:              {
3302:                 temp = (xend & 0x3ff) - (xinit & 0x3ff);
3303:                 if (temp <= 5)
3304:                    XEdge[yc][i].edge |= BRANCHFLAG;
3305:                 if ( xinit <= xref && xref <= xend )
3306:                 {
3307:                    // count the cross point
3308:                    if ( maxflag )
3309:                    {
3310:                       *xmax = temp;
3311:                       Leaf[leafindex].ob id[index] |= LEAVESFLAG;
3312:                    }
3313:                    *(ycross + icount) += 1;
3314:                 }
3315:              }
3316:           }
3317:        }
3318:     }
3319: }
3320:
3321: int CRunLength::SmallBranch ( int leaftype, int sindex, int count, LINE
     *BRLINE, int order )
3322: {
3323:    int BRCHAIN[10][10];
3324:    int YCOOR[10], cindex, ycent, first;
3325:    int temp1, temp2, next, yc, branch, ret;
3326:
3327:    yc = 0; count = 0;
3328:    branch = 0; first = 1;
3329:    CVt52App *PApp = (CVt52App *) AfxGetApp ();
3330:
3331:    BRLINE->x0 = -1;
3332:    BRLINE->y0 = -1;
3333:    BRLINE->x1 = -1;
3334:    BRLINE->y1 = -1;
3335:
3336:    for ( int j = 0 ; j < XRunRect.h ; j += m_sample, yc++ )
3337:    {
3338:       for ( int i = 0 ;; i++ )
3339:       {
3340:          next = 0;
3341:          temp1 = XEdge[yc][i].edge;
3342:          if ( temp1 == -1 ) break;
3343:
3344:          if ( temp1 & BRANCHFLAG )
3345:          {
3346:             if ( order == 0  && first )
3347:             {
3348:                BRLINE->x0 = (((temp1 & 0xff) +
3349:                   (XEdge[yc][i+1].edge & 0xff) ) >> 1) + XRunRect.x;
3350:                BRLINE->y0 = j + XRunRect.y;
3351:                first = 0;
3352:             }
3353:             else if ( order == 0 && first == 0)
3354:             {
3355:                temp2 = (((temp1 & 0xff) + (XEdge[yc][i+1].edge & 0xff))
```

```
3356:                        >> 1) + XRunRect.x;
3357:                  if ( abs (temp2 - BRLINE->x0) <= 5)
3358:                  {
3359:                     BRLINE->x1 = (short)temp2;
3360:                     BRLINE->y1 = (short) (j + XRunRect.y);
3361:                  }
3362:               }
3363:               if ( j < (XRunRect.h - 5) )
3364:               {
3365:                  for ( int k = 0 ;; k++ )
3366:                  {
3367:                     temp1 = XEdge[yc+1][k].edge;
3368:                     if ( temp1 == -1 ) break;
3369:
3370:                     if ( temp1 & BRANCHFLAG )
3371:                     {
3372:                        next = 1;
3373:                        break;
3374:                     }
3375:                  }
3376:               }
3377:            }
3378:
3379:            if ( next )
3380:            {
3381:               YCOOR[branch] = j;
3382:               ret = SearchChain ( yc, i, &BRCHAIN[branch][0], order );
3383:               if ( ret ) branch++;
3384:            }
3385:         }
3386:      }
3387:
3388:      int x0, x1, y0, y1;
3389:      int cid, nid, mod, index;
3390:      int brpos, i;
3391:      int min, lref, rref, cref;
3392:      int brid, color, init, NEWLEAF[10];
3393:
3394:      if ( branch && Leaf [leafindex].LEAF )
3395:      {
3396:         min = 800;
3397:         cref = Leaf [leafindex].xcent;
3398:         lref = Leaf [leafindex].xinit * m_sample + m_xorg;
3399:         rref = Leaf [leafindex].xend * m_sample + m_xorg;
3400:
3401:         for ( i = 0 ; i < branch ; i++ )
3402:         {
3403:            temp1 = BRCHAIN[i][0];
3404:            if ( temp1 < lref || temp1 > rref )
3405:               NEWLEAF[i] = 1;
3406:            else NEWLEAF[i] = 0;
3407:
3408:            if ( leaftype == YLARGE )
3409:            {
3410:               temp1 = abs( BRCHAIN[i][0] - cref );
3411:               temp2 = 1000;
3412:            }
3413:            else
3414:            {
3415:               temp1 = abs( BRCHAIN[i][0] - lref );
3416:               temp2 = abs( BRCHAIN[i][0] - rref );
3417:            }
3418:
3419:            if ( ( temp1 > temp2 ) && ( min > temp2 ) )
3420:            {
3421:               Leaf[leafindex].xbrpos = BRCHAIN[i][0];
3422:               min = temp2;
3423:               brid = i;
3424:            }
3425:            else if ( ( temp1 <= temp2 ) && ( min > temp1 ) )
3426:            {
3427:               Leaf[leafindex].xbrpos = BRCHAIN[i][0];
3428:               min = temp1;
3429:               brid = i;
3430:            }
```

```
3431:           else;
3432:
3433:       }
3434:    }
3435:
3436:    for ( j = 0 ; j < branch ; j++ )
3437:    {
3438:       if ( j == brid ) color = CDE_PEN_ORANGE;
3439:       else             color = CDE_PEN_BLUE;
3440:       ycent = 0;
3441:       for ( i = 1 ;; i++ )
3442:       {
3443:          if ( BRCHAIN [j][i] == -1 ) break;
3444:          if ( i > 1 ) ycent += m_sample;
3445:       }
3446:       ycent = ycent >> 1;
3447:       ycent += XRunRect.y + YCOOR[j];
3448:
3449:       if ( ycent >= Leaf[leafindex].ycent )
3450:            brpos = BRPOSLOW;
3451:       else  brpos = BRPOSHIGH;
3452:
3453:       for ( i = 1 ;; i++ )
3454:       {
3455:          x0 = BRCHAIN[j][i];
3456:          x1 = BRCHAIN[j][i+1];
3457:          if ( x0 == -1 || x1 == -1) break;
3458:
3459:          y0 = YCOOR[j] + XRunRect.y + m sample * (i-1);
3460:          y1 = y0 + m sample;
3461:
3462:          if (vdebug)
3463:             CIC draw line ( oh Console[PApp->Frame Index],
3464:                (double) x0, (double) y0, (double) x1,
3465:                (double) y1, color );
3466:          if ( brpos == BRPOSLOW  &&  i == 1 )
3467:          {
3468:             Leaf [leafindex].ybrpos = y0;
3469:             index = (x0 - m xorg) / m sample;
3470:             init = leafinit[leafindex] & 0x3ff;
3471:             if ( index >= init && index <= leafend [leafindex] )
3472:             {
3473:                cindex = index - init;
3474:                mod = x0 % m sample;
3475:                cid = Leaf [leafindex].ob id[sindex+cindex];
3476:                nid = Leaf [leafindex].ob id[sindex+cindex+1];
3477:                if (cid != -1 && nid != -1)
3478:                {
3479:                   temp1 = CANOBject[index][cid&0xff].redge;
3480:                   temp2 = CANOBject[index+1][nid&0xff].redge;
3481:                   temp1 = temp1 + (temp2 - temp1) * mod / m sample;
3482:                   if ( y0 <= (temp1 + m_sample + m_yorg)
3483:                      && NEWLEAF [j] )
3484:                   {
3485:                      if ( mod > 2 )
3486:                          AnotherLeaf ( index+1, nid & 0xff );
3487:                      else AnotherLeaf ( index, cid & 0xff );
3488:
3489:                      if (vdebug)
3490:                         CIC draw line ( oh Console[PApp->Frame Index],
3491:                            (double) x0, (double) (temp1+m yorg),
3492:                            (double) x0, (double) y0, color);
3493:                   }
3494:                }
3495:             }
3496:          }
3497:       }
3498:
3499:       if ( brpos == BRPOSHIGH )
3500:       {
3501:          x0 = BRCHAIN[j][i-1];
3502:          x1 = BRCHAIN[j][i];
3503:          index = (x1 - m xorg) / m sample;
3504:          init = leafinit[leafindex] & 0x3ff;
3505:          if  ( index >= init && index <= leafend [leafindex] )
```

```
3506:           {
3507:              Leaf [leafindex].ybrpos = y1;
3508:              cindex = index - init;
3509:              mod = x1 % m_sample;
3510:              cid = Leaf [leafindex].ob_id[sindex+cindex];
3511:              nid = Leaf [leafindex].ob_id[sindex+cindex+1];
3512:              if (cid != -1 && nid != -1)
3513:              {
3514:                 temp1 = CANObject[index][cid&0xff].fedge;
3515:                 temp2 = CANObject[index+1][nid&0xff].fedge;
3516:                 temp1 = temp1 + (temp2 - temp1) * mod / m_sample;
3517:                 if ( y1 >= (temp1 - m_sample + m_yorg) &&
3518:                    NEWLEAF [j] )
3519:                 {
3520:                    if ( mod > 2 )
3521:                        AnotherLeaf ( index+1, nid & 0xff );
3522:                    else AnotherLeaf ( index, cid & 0xff );
3523:                    if (vdebug)
3524:                       CIC_draw_line ( oh_Console[PApp->Frame_Index],
3525:                          (double) x1, (double) (temp1+m_yorg),
3526:                          (double) x1, (double) y1, color );
3527:                 }
3528:              }
3529:           }
3530:        }
3531:     }
3532:
3533:     if ( branch ) return 1;
3534:     else          return 0;
3535:
3536: }
3537:
3538: void CRunLength::AnotherLeaf ( int index, int obid )
3539: {
3540:     int next_id, first, count;
3541:     int cindex, cid;
3542:     int Complete, eindex, temp1;
3543:     CVt52App *PApp = (CVt52App *) AfxGetApp ();
3544:
3545:     cindex = index;
3546:     cid = obid;
3547:     first = 0;
3548:     Complete = 0;
3549:     for ( ;; cindex++ )
3550:     {
3551:        next_id = CANObject[cindex][cid].next_id;
3552:        if (next_id == -1)
3553:        {
3554:           if ( first == 1 )
3555:              eindex = cindex;
3556:           break;
3557:        }
3558:        temp1 = CANObject[cindex][cid].leaf;
3559:        if ( temp1 == 1 )
3560:        {
3561:           Complete = 1;
3562:           first = 1;
3563:           count = 0;
3564:        }
3565:        if ( first == 1 && temp1 == 0 )
3566:        {
3567:           if ( CANObject[cindex][cid].length <
3568:               CANObject[cindex][next_id&0xff].length &&
3569:               CANObject[cindex][cid].length < 10 )
3570:           {
3571:               eindex = cindex;
3572:               break;
3573:           }
3574:        }
3575:        cid = next_id & 0xff;
3576:     }
3577:
3578:     int xc;
3579:
3580:     if ( Complete )
```

```
3581:     {
3582:        xc = index * m_sample + m_xorg;
3583:        cid = obid;
3584:        for ( cindex = index ; cindex < (eindex - 1); cindex++, xc += m_sample )
3585:        {
3586:           next_id = CANObject[cindex][cid].next_id;
3587:           CIC_draw_line ( oh_Console[PApp->Frame_Index],
3588:              (double) (xc), (double) (CANObject[cindex][cid].fedge+m_yorg),
3589:              (double) (xc + m_sample),
3590:              (double) (CANObject[cindex+1][next_id&0xff].fedge+m_yorg),
      CDE_PEN_RED );
3591:
3592:           CIC_draw_line ( oh_Console[PApp->Frame_Index],
3593:              (double) (xc), (double) (CANObject[cindex][cid].redge+m_yorg),
3594:              (double) (xc + m_sample),
3595:              (double) (CANObject[cindex+1][next_id&0xff].redge+m_yorg),
      CDE_PEN_RED );
3596:           cid = next_id & 0xff;
3597:        }
3598:     }
3599: }
3600: int CRunLength::SearchChain ( int yc, int i, int *BRANCH, int order)
3601: {
3602:    int k, chain, xavg;
3603:    int xcent1, xcent2, temp1, temp2;
3604:    int VALIDCHAIN, init;
3605:
3606:    xcent1 = (XEdge [yc][i+1].edge&0xff) + (XEdge [yc][i].edge&0xff);
3607:    xavg = xcent1;
3608:    chain = 0;
3609:    yc++;
3610:    k = yc * m_sample;
3611:    for ( ; k <= XRunRect.h ; k += m_sample, yc++ )
3612:    {
3613:       VALIDCHAIN = 0;
3614:       for ( int kkk = 0 ;; kkk++ )
3615:       {
3616:          temp1 = XEdge[yc][kkk].edge;
3617:          temp2 = XEdge[yc][kkk+1].edge;
3618:          if ( temp1 == -1 || temp2 == -1 ) break;
3619:          xcent2 = (temp1 & 0xff) + (temp2 & 0xff);
3620:
3621:          if  ( (temp1 & BRANCHFLAG) && (abs(xcent2-xcent1) < 10) )
3622:          {
3623:             if (chain == 0)
3624:             {
3625:                XEdge[yc-1][i].edge &= ~BRANCHFLAG;
3626:                *(BRANCH+1) = (xcent1 >> 1) + XRunRect.x;
3627:                chain++;
3628:             }
3629:             *(BRANCH+chain+1) = (xcent2 >> 1) + XRunRect.x;
3630:             xavg += xcent2;
3631:             VALIDCHAIN = 1;
3632:             xcent1 = xcent2;
3633:             chain++;
3634:             XEdge[yc][kkk].edge &= ~BRANCHFLAG;
3635:             break;
3636:          }
3637:       }
3638:       if ( VALIDCHAIN == 0 ) break;
3639:    }
3640:
3641:    *(BRANCH + chain + 1) = -1;
3642:    if (chain)
3643:    {
3644:       xcent1  = ((xavg / chain) >> 1) + XRunRect.x;
3645:       *BRANCH = xcent1;
3646:       if (order)
3647:       {
3648:          init = leafinit[leafindex] & 0x3ff;
3649:          temp1 = ( init - 2 ) * m_sample + m_xorg;
3650:          temp2 = ( leafend[leafindex] + 2 ) * m_sample + m_xorg;
3651:          if ( xcent1 < temp1 || xcent1 > temp2 )
3652:             return 0;
3653:       }
```

```
3654:     }
3655:
3656:     if (chain) return 1;
3657:     else       return 0;
3658: }
3659:
3660: void CRunLength::SearchBranch ( int leaftype, int xindex, int sindex, int
      icount )
3661: {
3662:     LINE BRLINE;
3663:     CACQ rect LRECT, RRECT, TRECT;
3664:     CVt52App *PApp = (CVt52App *) AfxGetApp ();
3665:     int temp, ret;
3666:     int cid, fedge, redge;
3667:
3668:     switch ( leaftype )
3669:     {
3670:        case YLARGE:
3671:           LRECT.x = XRunRect.x + 15;
3672:           LRECT.y = XRunRect.y - 10;
3673:           LRECT.w = XRunRect.w - 30;
3674:           LRECT.h = 30;
3675:        break;
3676:        case XLARGE:
3677:        case XYEQUAL:
3678:           cid = Leaf[leafindex].ob id[sindex];
3679:           if ( cid != -1 )
3680:           {
3681:              fedge = CANOBject[xindex][cid&0xff].fedge;
3682:              redge = CANOBject[xindex][cid&0xff].redge;
3683:
3684:              LRECT.x = xindex * m sample + m xorg - 10;
3685:              LRECT.y = fedge + m yorg - 15;
3686:              LRECT.w = 20;
3687:              LRECT.h = redge - fedge  + 30;
3688:           }
3689:        break;
3690:     }
3691:
3692:     RECTSave ( &TRECT );
3693:     RECTRestore ( LRECT );
3694:        XScan ( 1 );
3695:        ret = SmallBranch ( leaftype, sindex, icount, &BRLINE, 0 );
3696:     RECTRestore ( TRECT );
3697:
3698:     if ( vdebug )
3699:     {
3700:        CIC draw rect ( oh Console[PApp->Frame Index], (double) LRECT.x,
3701:           (double) LRECT.y, (double) LRECT.w, (double) LRECT.h,
3702:                 CDE PEN ORANGE );
3703:        if (ret == 0) Draw_Branch ( BRLINE );
3704:     }
3705:
3706:     switch ( leaftype )
3707:     {
3708:        case YLARGE:
3709:           RRECT.x = XRunRect.x + 15;
3710:           RRECT.y = XRunRect.y + XRunRect.h - 5 - 15;
3711:           RRECT.w = XRunRect.w - 30;
3712:           RRECT.h = 30;
3713:        break;
3714:        case XLARGE:
3715:        case XYEQUAL:
3716:           temp = icount - 1;
3717:           cid = Leaf[leafindex].ob id[sindex+temp];
3718:           if ( cid != -1 )
3719:           {
3720:              fedge = CANOBject[xindex+temp][cid&0xff].fedge;
3721:              redge = CANOBject[xindex+temp][cid&0xff].redge;
3722:
3723:              RRECT.x = (xindex+temp) * m sample + m xorg - 10;
3724:              RRECT.y = fedge + m yorg - 15;
3725:              RRECT.w = 20;
3726:              RRECT.h = redge - fedge  + 30;
3727:           }
```

```
3728:        break;
3729:     }
3730:
3731:     RECTSave ( &TRECT );
3732:     RECTRestore ( RRECT );
3733:        XScan ( 1 );
3734:        ret = SmallBranch ( leaftype, sindex, icount, &BRLINE, 0 );
3735:     RECTRestore ( TRECT );
3736:
3737:     if ( vdebug )
3738:     {
3739:        CIC draw rect ( oh Console[PApp->Frame Index], (double) RRECT.x,
3740:           (double) RRECT.y, (double) RRECT.w, (double) RRECT.h,
3741:                 CDE PEN ORANGE );
3742:        if ( ret == 0 && Leaf[leafindex].LEAF )
3743:           Draw Branch ( BRLINE );
3744:     }
3745: }
3746:
3747: void CRunLength::RECTSave( CACQ_rect *mrect )
3748: {
3749:
3750:     mrect->x = XRunRect.x;
3751:     mrect->y = XRunRect.y;
3752:     mrect->w = XRunRect.w;
3753:     mrect->h = XRunRect.h;
3754: }
3755:
3756: void CRunLength::RECTRestore ( CACQ_rect mrect )
3757: {
3758:     XRunRect.x = mrect.x;
3759:     XRunRect.y = mrect.y;
3760:     XRunRect.w = mrect.w;
3761:     XRunRect.h = mrect.h;
3762: }
3763:
3764: void CRunLength::MainBranch ()
3765: {
3766:     int next id, xc, y0 cent, y1_cent;
3767:     int i, j, first, ret;
3768:     CVt52App *PApp = (CVt52App *) AfxGetApp ();
3769:
3770:     xc = 0;
3771:     first = 1;
3772:     for ( i = 0 ; i < (xcount-1) ; i++, xc += m_sample )
3773:     {
3774:        if ( first == 1 )
3775:        {
3776:           for ( j = 0 ; j < OBCount[i] ; j++)
3777:           {
3778:              next id = CANOBject[i][j].next id;
3779:              if ( CANOBject[i][j].type == NORMAL && next_id != -1 )
3780:              {
3781:                 first = 0;
3782:                 break;
3783:              }
3784:           }
3785:        }
3786:
3787:        if ( first == 0 )
3788:        {
3789:           next id = CANOBject[i][j].next id;
3790:           if ( CANOBject[i][j].type == NORMAL && next_id != -1 )
3791:           {
3792:              next id &= 0xff;
3793:              CANOBject[i][j].type = MAINBRANCH;
3794:              y0 cent = ( CANOBject[i][j].fedge + CANOBject[i][j].redge ) >> 1;
3795:              y1 cent = ( CANOBject[i+1][next id].fedge +
3796:                       CANOBject[i+1][next id].redge ) >> 1;
3797:
3798:              j = next id;
3799:              if (vdebug)
3800:                 CIC draw line ( oh Console[PApp->Frame Index], (double)
      (xc+m xorg),
3801:                       (double) (y0_cent+m_yorg), (double) (xc+m_xorg+m_sample),
```

```
3802:                     (double) (y1_cent+m_yorg), CDE_PEN_ORANGE );
3803:          }
3804:          else if ( next_id == -1 )
3805:          {
3806:             CANOBject[i][j].type = MAINBRANCH;
3807:             ret = Next Search ( &i, &j );
3808:             if ( ret == 0 ) break;
3809:             i--;
3810:             xc = i * m sample;
3811:          }
3812:       }
3813:    }
3814: }
3815:
3816: int CRunLength::Next_Search ( int *i, int *j )
3817: {
3818:    int y0 cent, y1 cent, k;
3819:    int diff, min, minpos, next_id;
3820:
3821:    y0 cent = CANOBject[*i][*j].fedge + CANOBject[*i][*j].redge;
3822:    *i += 1;
3823:    min = 1000;
3824:
3825:    for ( ; *i < (xcount-1) ; *i += 1 )
3826:    {
3827:       for ( k = 0 ; k < OBCount [*i] ; k++ )
3828:       {
3829:          next id = CANOBject[*i][k].next id;
3830:          if ( CANOBject[*i][k].type == NORMAL && next_id != -1 &&
3831:              CANOBject[*i][k].prev id == -1 )
3832:          {
3833:             y1 cent = CANOBject[*i][k].fedge +
3834:                    CANOBject[*i][k].redge;
3835:             diff = abs (y1 cent - y0 cent);
3836:             if ( ( min > diff ) && ( diff <= 40 ) )
3837:             {
3838:                if ( CheckLeaf ( *i, k ) == 0 )
3839:                {
3840:                   min = diff;
3841:                   minpos = k;
3842:                }
3843:             }
3844:
3845:          }
3846:       }
3847:
3848:       if ( min != 1000 )
3849:       {
3850:          *j = minpos;
3851:          return 1;
3852:       }
3853:    }
3854:
3855:    return 0;
3856: }
3857:
3858: int CRunLength::CheckLeaf ( int i, int ob_id )
3859: {
3860:    int init, cid;
3861:
3862:    for ( int k = 0 ; k < rcount ; k++)
3863:    {
3864:       init = leafinit[k];
3865:       if ( (init & 0x3ff) == i  && ( init & LEAFINITFLAG ) )
3866:       {
3867:          cid = Leaf[k].ob id[0] & 0xff;
3868:          if ( ob id == cid )
3869:             return 1;
3870:       }
3871:    }
3872:
3873:    return 0;
3874: }
3875:
3876: void CRunLength::Draw_Branch ( LINE BRLINE )
```

```
3877: {
3878:     int index, ycent, temp, mod;
3879:     int xbrcent, ybrcent, ybrnext, nid;
3880:     int diff;
3881:     CVt52App *PApp = (CVt52App *) AfxGetApp ();
3882:
3883:     if ( BRLINE.x0 == -1 && BRLINE.x1 == -1) return;
3884:     else if ( BRLINE.x0 != -1 && BRLINE.x1 != -1)
3885:     {
3886:        temp = (BRLINE.x0 + BRLINE.x1) >> 1;
3887:        ycent = (BRLINE.y0 + BRLINE.y1) >> 1;
3888:     }
3889:     else if ( BRLINE.x0 != -1)
3890:     {
3891:        temp = BRLINE.x0;
3892:        ycent = BRLINE.y0;
3893:     }
3894:
3895:     xbrcent =  temp;
3896:     ybrcent = 0;
3897:     index = ( temp - m_xorg ) / m_sample;
3898:     mod = ( temp - m_xorg ) % m_sample;
3899:
3900:     for ( int j = 0 ; j < OBCount[index] ; j++ )
3901:     {
3902:        if ( CANObject[index][j].type == MAINBRANCH )
3903:        {
3904:           nid = CANObject[index][j].next_id;
3905:           if ( nid != -1 )
3906:           {
3907:              nid &= 0xff;
3908:              ybrcent = CANObject[index][j].fedge +
3909:                      CANObject[index][j].redge;
3910:              ybrnext = CANObject[index+1][nid].fedge +
3911:                      CANObject[index+1][nid].redge;
3912:              diff = ybrnext - ybrcent;
3913:              ybrcent >>= 1;
3914:
3915:              ybrcent += ( (diff * mod / m_sample) >> 1 );
3916:           }
3917:        }
3918:     }
3919:
3920:     if ( ybrcent == 0 )
3921:        ybrcent = CALBRCenter ( index, mod );
3922:
3923:     if ( vdebug )
3924:        CIC_draw_line ( oh_Console[PApp->Frame_Index], (double) (xbrcent),
3925:           (double) (ycent), (double) (xbrcent),
3926:           (double) (ybrcent+m_yorg), CDE_PEN_ORANGE );
3927: }
3928:
3929: int CRunLength::CALBRCenter ( int index, int mod )
3930: {
3931:     int i1,y1;
3932:     int i2,y2;
3933:     int temp;
3934:     int ycent, diff;
3935:
3936:     i2 = index + 1;
3937:     y2 = 0;
3938:     for (  ; i2 < xcount ; i2++ )
3939:     {
3940:        for ( int j = 0 ; j < OBCount[i2] ; j++ )
3941:           if ( CANObject[i2][j].type == MAINBRANCH )
3942:           {
3943:              y2 = CANObject[i2][j].fedge + CANObject[i2][j].redge;
3944:              break;
3945:           }
3946:        if (y2) break;
3947:     }
3948:
3949:     i1 = index - 1;
3950:     for ( ; i1 > 0 ; i1--)
3951:     {
```

```
3952:        for ( int j = 0 ; j < OBCount[i1] ; j++ )
3953:           if ( CANOBject[i1][j].type == MAINBRANCH )
3954:           {
3955:              y1 = CANOBject[i1][j].fedge + CANOBject[i1][j].redge;
3956:              break;
3957:           }
3958:        if (y1) break;
3959:     }
3960:
3961:     diff = y2 - y1;
3962:     ycent = y1 >> 1;
3963:     temp = (i2 - index) * m_sample + mod;
3964:     temp = temp * diff / ((i2 - i1) * m_sample);
3965:     ycent += (temp >> 1);
3966:
3967:     return ycent;
3968: }
3969:
3970: void CRunLength::FindCrossPoint ()
3971: {
3972:     int i, j, k, xc, yc;
3973:     int nid, pid, jcount, kcount;
3974:     CVt52App *PApp = (CVt52App *) AfxGetApp ();
3975:
3976:     for ( i = 0, xc = m_xorg ; i < xcount ; i++, xc += m_sample )
3977:     {
3978:        for ( j = 0 ; j < OBCount [i] ; j++ )
3979:        {
3980:           nid = CANOBject [i][j].next_id;
3981:           pid = CANOBject [i][j].prev_id;
3982:
3983:           if ( nid != -1 || pid != -1 )
3984:           {
3985:              for ( k = 0 ; k < OBCount [i] ; k++ )
3986:              {
3987:                 if ( j != k )
3988:                 {
3989:
3990:                    if ( nid == CANOBject[i][k].next_id )
3991:                    {
3992:                       jcount = CANOBject[i][j].count;
3993:                       kcount = CANOBject[i][k].count;
3994:                       if ( jcount >= 1 && kcount >= 1 )
3995:                       {
3996:                          CANOBject[i+1][nid].cross = OBCROSS;
3997:                          if ( jcount < 10 || kcount < 10 )
3998:                             FindBranches ( i, j, k, 0 );
3999:                          if (vdebug)
4000:                          {
4001:                             yc = ( CANOBject[i+1][nid].fedge +
4002:                                   CANOBject[i+1][nid].redge ) >> 1;
4003:                             CIC draw cross ( oh Console[PApp->Frame Index],
4004:                                (double) (xc+m_sample),
4005:                                (double) (yc+m_yorg), 5, 0 ,
4006:                                CDE_PEN_ORANGE );
4007:                          }
4008:                       }
4009:                    }
4010:
4011:                    if ( pid == CANOBject[i][k].prev_id )
4012:                    {
4013:                       jcount = CANOBject[i][j].count;
4014:                       kcount = CANOBject[i][k].count;
4015:                       if ( jcount >= 1 && kcount >= 1 )
4016:                       {
4017:                          CANOBject[i-1][pid].cross = OBCROSS;
4018:                          if ( jcount < 10 || kcount < 10 )
4019:                             FindBranches ( i, j, k, 1 );
4020:                          if (vdebug)
4021:                          {
4022:                             yc = ( CANOBject[i-1][pid].fedge +
4023:                                   CANOBject[i-1][pid].redge ) >> 1;
4024:                             CIC draw cross ( oh Console[PApp->Frame Index],
4025:                                (double) (xc-m_sample),
4026:                                (double) (yc+m_yorg), 5, 0,
```

```
4027:                                CDE PEN ORANGE );
4028:                            }
4029:                        }
4030:                    }
4031:                }
4032:            }
4033:        }
4034:    }
4035: }
4036:
4037:
4038: void CRunLength::FindBranches ( int i, int j, int k, int type )
4039: {
4040:    int cid, pid, xc, jcount, kcount;
4041:    int index, cfedge, credge, pfedge, predge;
4042:    int cross;
4043:    CVt52App *PApp = (CVt52App *) AfxGetApp ();
4044:
4045:    index = i;
4046:    jcount = CANOBject [index][j].count;
4047:    kcount = CANOBject [index][k].count;
4048:
4049:    if ( type )
4050:    {
4051:    }
4052:    else
4053:    {
4054:       if ( jcount > kcount )
4055:       {
4056:          cid = k;
4057:          pid = CANOBject [index][k].prev id;
4058:       }
4059:       else
4060:       {
4061:          cid = j;
4062:          pid = CANOBject [index][j].prev id;
4063:       }
4064:
4065:       xc = index * m sample + m xorg;
4066:       for ( ; index > 1 ; index--, xc -= m_sample )
4067:       {
4068:          CANOBject [index][cid].type = LEAFNODE;
4069:          cross = CANOBject [index][cid].cross;
4070:          if ( cross == OBCHILD || cross == OBPARENT )
4071:             CANOBject [index][cid].cross += OBCHECK;
4072:             //ClearCrossOBject ( index );
4073:
4074:          if ( pid == -1 ) break;
4075:
4076:          cfedge = CANOBject [index][cid].fedge;
4077:          credge = CANOBject [index][cid].redge;
4078:
4079:          pfedge = CANOBject [index-1][pid].fedge;
4080:          predge = CANOBject [index-1][pid].redge;
4081:
4082:          cid = pid;
4083:          pid = CANOBject [index-1][pid].prev id;
4084:
4085:          if ( vdebug )
4086:          {
4087:             CIC draw line ( oh Console[PApp->Frame Index],
4088:                (double) xc, (double) ( cfedge+m_yorg ),
4089:                (double) ( xc-m sample ),
4090:                (double) ( pfedge+m yorg ), CDE PEN RED );
4091:             CIC draw line ( oh Console[PApp->Frame Index],
4092:                (double) xc, (double) ( credge+m_yorg ),
4093:                (double) ( xc-m sample ),
4094:                (double) ( predge+m_yorg ), CDE_PEN_RED );
4095:          }
4096:       }
4097:    }
4098: }
4099:
4100: void CRunLength::ClearCrossOBject ( int index )
```

```
4102: {
4103:    int j, cross;
4104:
4105:    for ( j = 0 ; j < OBCount [index] ; j++ )
4106:    {
4107:       cross = CANObject [index][j].cross;
4108:       if ( cross == OBPARENT || cross == OBCHILD )
4109:          CANObject [index][j].cross += OBCHECK;
4110:    }
4111: }
4112:
4113: void CRunLength::FindRelinkPoint ()
4114: {
4115:    int i, j, pid, cid, nid;
4116:    int diff, min, minid, ret;
4117:    int cavg, pavg, fedge, redge;
4118:    int xc, yc, pfedge, predge;
4119:    int crossindex, crossid, CROSSUPDATE;
4120:    CVt52App *PApp = (CVt52App *) AfxGetApp ();
4121:
4122:    i = relinkindex;
4123:    cid = relinkid;
4124:    pid = CANObject [i][relinkid].prev id;
4125:    CROSSUPDATE = 0;
4126:    for ( ; i > 0 ; i-- )
4127:    {
4128:       if ( CANObject [i][cid].cross == OBCROSS )
4129:       {
4130:          crossindex = i;
4131:          crossid = cid;
4132:          CROSSUPDATE = 1;
4133:       }
4134:       if ( ( pid == -1 ) && CROSSUPDATE )
4135:       {
4136:          fedge = CANObject [i][cid].fedge;
4137:          redge = CANObject [i][cid].redge;
4138:          cavg = fedge + redge;
4139:          min = 1000;
4140:          for ( j = 0 ; j < OBCount[i-1] ; j++ )
4141:          {
4142:             pfedge = CANObject [i-1][j].fedge;
4143:             predge = CANObject [i-1][j].redge;
4144:             pavg = pfedge + predge;
4145:             diff = abs ( pavg - cavg );
4146:             if ( min > diff )
4147:             {
4148:                min = diff;
4149:                minid = j;
4150:             }
4151:          }
4152:
4153:          pfedge = CANObject [i-1][minid].fedge;
4154:          predge = CANObject [i-1][minid].redge;
4155:
4156:          if ( ( pfedge <= fedge && predge >= fedge ) ||
4157:               ( pfedge <= redge && predge >= redge ) ||
4158:               ( pfedge <= fedge && predge >= redge ) )
4159:          {
4160:             CANObject [i-1][minid].cross = OBCROSS;
4161:             CANObject [i-1][minid].next id = cid;
4162:             CANObject [i][cid].prev id = minid;
4163:             if ( vdebug )
4164:             {
4165:                xc = (i - 1) * m sample + m xorg;
4166:                yc = (( pfedge + predge ) >> 1) + m yorg;
4167:                CIC draw cross ( oh Console[PApp->Frame Index],
4168:                   (double) xc, (double) yc, 5, 0,  CDE PEN ORANGE );
4169:                CIC draw line ( oh Console[PApp->Frame Index],
4170:                   (double) xc, (double) yc,(double) ( xc+m sample ),
4171:                (double) ( (cavg>>1)+m_yorg ), CDE_PEN_ORANGE );
4172:             }
4173:             nid = cid;
4174:             for ( j = i ; j < crossindex ; j++ )
4175:             {
4176:                CANObject [j][nid].type = NORMAL;
```

```
4177:                nid = CANOBject [j][nid].next id;
4178:             }
4179:          }
4180:          else
4181:          {
4182:             if ( CROSSUPDATE && abs ( crossindex - i ) < 20 )
4183:             {
4184:                CROSSUPDATE = 0;
4185:                ret = NextSearch ( crossindex-1, crossid, cid );
4186:                if ( ret != -1 )
4187:                {
4188:                   i = crossindex-1;
4189:                   pid = ret;
4190:                }
4191:                else
4192:                   break;
4193:             }
4194:          }
4195:       }
4196:       cid = pid;
4197:       pid = CANOBject [i-1][pid].prev id;
4198:    }
4199:
4200: }
4201:
4202: int  CRunLength::NextSearch ( int index, int crossid, int cid )
4203: {
4204:    int i;
4205:
4206:    for ( i = 0 ; i < OBCount[index] ; i++ )
4207:    {
4208:       if ( i != cid )
4209:          if ( CANOBject [index][i].next_id == crossid )
4210:             return i;
4211:    }
4212:    return -1;
4213: }
4214: void CRunLength::FindIslandLeaf ()
4215: {
4216:    int i, j;
4217:    int type, count, next_id;
4218:
4219:    for ( i = 0 ; i < xcount ; i++ )
4220:    {
4221:       for ( j = 0 ; j < OBCount[i] ; j++ )
4222:       {
4223:          next id = CANOBject [i][j].next id;
4224:          if ( next_id == -1 )
4225:          {
4226:             type = CANOBject [i][j].type;
4227:             count = CANOBject [i][j].count;
4228:
4229:             if ( ( type == NORMAL || type == LEAFNODE ) &&
4230:                   count < 10 && count >= 1 )
4231:                LinkIslandLeaf ( i, j );
4232:          }
4233:       }
4234:    }
4235: }
4236:
4237: void CRunLength::LinkIslandLeaf ( int i, int j )
4238: {
4239:    int index, iid, cid, eid;
4240:    int mindex, mid, max, temp, cross;
4241:    int iindex, eindex, pid, ret, tempid;
4242:    int imin, mmin, emin, xnewmin, newid;
4243:    CVt52App *PApp = (CVt52App *) AfxGetApp ();
4244:
4245:    index = i;
4246:    cid = j;
4247:    eindex = i;
4248:    eid = j;
4249:
4250:    max = 0;
4251:    cross = 0;
```

```
4252:     mindex = -1;
4253:     for ( ; index > 0 ; index-- )
4254:     {
4255:        temp = CANOBject [index][cid].length;
4256:        cross += CANOBject [index][cid].cross;
4257:        if ( temp > max )
4258:        {
4259:           max = temp;
4260:           mindex = index;
4261:           mid = cid;
4262:        }
4263:
4264:        pid = CANOBject [index][cid].prev id;
4265:
4266:        if ( pid == -1 )
4267:        {
4268:           iindex = index;
4269:           iid = cid;
4270:           break;
4271:        }
4272:
4273:        cid = pid;
4274:     }
4275:
4276:     if (index == 0)
4277:     {
4278:        iindex = index;
4279:        iid = cid;
4280:     }
4281:
4282:     ret = -1;
4283:     mmin = 500;
4284:     if (mindex != -1)
4285:        ret = NearObject ( mindex, mid, cross, &mmin, &xnewmin, &newid, 0 );
4286:     ret = NearObject ( iindex, iid, cross, &imin, &xnewmin, &newid, 0 );
4287:     ret = NearObject ( eindex, eid, cross, &emin, &xnewmin, &newid, 0 );
4288:
4289:     ret = 1;
4290:     if ( imin >= mmin && emin >= mmin && mmin <= 4 )
4291:     {
4292:        ret = NearObject ( mindex, mid, cross, &mmin, &xnewmin, &newid, 1 );
4293:        if ( ret == 0 )
4294:        {
4295:           mindex = xnewmin;
4296:           mid = newid;
4297:           MaskLeaf ( iindex, iid, eindex, eid );
4298:           ret = NearObject ( mindex, mid, cross, &mmin, &xnewmin, &newid, 1 );
4299:           FreeMask ( iindex, iid, eindex, eid );
4300:        }
4301:     }
4302:     else if ( mmin >= imin && emin > imin && imin <= 4 )
4303:     {
4304:        ret = NearObject ( iindex, iid, cross, &imin, &xnewmin, &newid, 1 );
4305:        if ( ret == 0 )
4306:        {
4307:           temp = xnewmin;
4308:           tempid = newid;
4309:           MaskLeaf ( iindex, iid, eindex, eid );
4310:           ret = NearObject ( temp, tempid, cross, &imin, &xnewmin, &newid, 1 );
4311:           FreeMask ( iindex, iid, eindex, eid );
4312:        }
4313:     }
4314:     else if ( mmin >= emin && imin > emin && emin <= 4)
4315:     {
4316:        ret = NearObject ( eindex, eid, cross, &emin, &xnewmin, &newid, 1 );
4317:        if ( ret == 0 )
4318:        {
4319:           temp = xnewmin;
4320:           tempid = newid;
4321:           MaskLeaf ( iindex, iid, eindex, eid );
4322:           ret = NearObject ( eindex, eid, cross, &emin, &xnewmin, &newid, 1 );
4323:           FreeMask ( iindex, iid, eindex, eid );
4324:        }
4325:     }
4326:     else
```

```
4327:     {
4328:        LastDecideLeaf ( iindex, iid, mindex, mid, eindex, eid );
4329:        ret = 0;
4330:     }
4331:
4332:     if ( ret )
4333:        DrawLeaf ( iindex, iid, eindex, eid, CDE PEN RED );
4334: }
4335: void CRunLength::MaskLeaf ( int init, int iid, int end, int eid )
4336: {
4337:     int i, cid;
4338:
4339:     cid = iid;
4340:     for ( i = init ; i <= end ; i++ )
4341:     {
4342:        if ( cid == -1 ) break;
4343:        CANOBject[i][cid].type |= LEAVESMASK;
4344:        cid = CANOBject[i][cid].next id;
4345:     }
4346: }
4347:
4348: void CRunLength::FreeMask ( int init, int iid, int end, int eid )
4349: {
4350:     int i, cid;
4351:
4352:     cid = iid;
4353:     for ( i = init ; i <= end ; i++ )
4354:     {
4355:        if ( cid == -1 ) break;
4356:        CANOBject[i][cid].type &= 0xff;
4357:        cid = CANOBject[i][cid].next id;
4358:     }
4359: }
4360: int CRunLength::NearObject ( int mindex, int mid, int cross, int *diff,
4361:                              int *xminindex, int *minid, int type )
4362: {
4363:     int i, j, diff1, diff2, pid, nid;
4364:     int min, minpos, minindex, temp1, temp2;
4365:     int fedge, redge;
4366:     CVt52App *PApp = (CVt52App *) AfxGetApp ();
4367:
4368:
4369:     fedge = CANOBject [mindex][mid].fedge;
4370:     redge = CANOBject [mindex][mid].redge;
4371:     pid   = CANOBject [mindex][mid].prev id;
4372:     nid   = CANOBject [mindex][mid].next id;
4373:
4374:     min = 10000;
4375:     for ( i = mindex-1 ; i < (mindex+2) ; i++ )
4376:     {
4377:        for ( j = 0 ; j < OBCount [i] ; j++ )
4378:        {
4379:           if ( CANOBject [i][j].type != REFINE &&
4380:                (CANOBject [i][j].type & LEAVESMASK) == 0 )
4381:           {
4382:              if ( (i == mindex && mid != j) || ( i == (mindex-1) && pid != j)
4383:                     || ( i == (mindex+1) && nid != j ) )
4384:              {
4385:                 temp1 = CANOBject [i][j].fedge;
4386:                 temp2 = CANOBject [i][j].redge;
4387:
4388:                 diff1 = abs (fedge - temp2);
4389:                 diff2 = abs (redge - temp1);
4390:                 if ( min > abs (diff1) || min > abs (diff2) )
4391:                 {
4392:                    if ( diff1 > diff2 ) min = diff2;
4393:                    else                 min = diff1;
4394:                    minpos = j;
4395:                    minindex = i;
4396:                 }
4397:              }
4398:           }
4399:        }
4400:     }
4401:
```

```
4402:     if ( type == 0 )
4403:     {
4404:        *diff = min;
4405:        return 1;
4406:     }
4407:     if ( min <= 4 )
4408:     {
4409:        if ( cross == 0 )
4410:        {
4411:           pid = CANObject [minindex][minpos].prev id;
4412:           nid = CANObject [minindex][minpos].next id;
4413:           if ( ( pid != -1 && nid == -1 ) ||
4414:                ( pid == -1 && nid != -1 ) )
4415:              AddCrossPoint ( minindex, minpos, mindex);
4416:           else if ( pid != -1 && nid != -1 )
4417:              CANObject [minindex][minpos].cross = OBCROSS;
4418:
4419:           temp1 = CANObject[minindex][minpos].fedge +
4420:                   CANObject[minindex][minpos].redge;
4421:           temp2 = CANObject[mindex][mid].fedge +
4422:                   CANObject[mindex][mid].redge;
4423:           i = minindex * m_sample + m_xorg;
4424:           j = mindex * m_sample + m_xorg;
4425:           if ( vdebug )
4426:           {
4427:              if ( pid != -1 && nid != -1 )
4428:                 CIC draw cross ( oh Console [PApp->Frame Index],
4429:                    (double) i, (double) ( (temp1>>1) + m_yorg ),
4430:                    5, 0, CDE PEN BLUE );
4431:              CIC draw line ( oh Console [PApp->Frame Index],
4432:                 (double) i, (double) ( ( temp1 >> 1 ) + m_yorg ),
4433:                 (double) j, (double) ( ( temp2 >> 1 ) + m_yorg ),
4434:                 CDE PEN RED );
4435:
4436:           }
4437:           if ( pid == -1 && nid == -1 )
4438:           {
4439:              *xminindex = minindex;
4440:              *minid     = minpos;
4441:              return 0;
4442:           }
4443:        }
4444:        return 1;
4445:     }
4446:     else
4447:        return -1;
4448: }
4449:
4450: void CRunLength::DrawLeaf ( int iindex, int iid, int eindex, int eid, int colo
     r )
4451: {
4452:     int xc, cid, nid;
4453:     int cross, index;
4454:     int cfedge, credge;
4455:     int nfedge, nredge;
4456:     CVt52App *PApp = (CVt52App *) AfxGetApp ();
4457:
4458:     xc = iindex * m_sample + m_xorg;
4459:     cid = iid;
4460:     for ( index = iindex ; index < eindex ; index++, xc += m_sample )
4461:     {
4462:        CANObject [index][cid].type = LEAFNODE;
4463:        cross = CANObject [index][cid].cross;
4464:        if ( cross == OBCHILD || cross == OBPARENT )
4465:           CANObject [index][cid].cross += OBCHECK;
4466:        // ClearCrossOBject ( index );
4467:        nid = CANObject [index][cid].next id;
4468:        if ( nid == -1 ) break;
4469:
4470:        cfedge = CANObject [index][cid].fedge;
4471:        credge = CANObject [index][cid].redge;
4472:
4473:        nfedge = CANObject [index+1][nid].fedge;
4474:        nredge = CANObject [index+1][nid].redge;
4475:
```

```
4476:        cid = nid;
4477:
4478:        if ( vdebug )
4479:        {
4480:           CIC draw line ( oh Console[PApp->Frame Index],
4481:              (double) xc, (double) ( cfedge+m_yorg ),
4482:              (double) ( xc+m sample ),
4483:              (double) ( nfedge+m yorg ), color );
4484:           CIC draw line ( oh Console[PApp->Frame Index],
4485:              (double) xc, (double) ( credge+m_yorg ),
4486:              (double) ( xc+m sample ),
4487:              (double) ( nredge+m_yorg ), color );
4488:        }
4489:     }
4490: }
4491:
4492: void CRunLength::LastDecideLeaf ( int iindex, int iid, int mindex, int mid,
4493:                                   int eindex, int eid )
4494: {
4495:     int i, sum, cid, length, type;
4496:     int gravg, ret;
4497:
4498:     sum = 0;
4499:     cid = iid;
4500:     length = 0;
4501:     for ( i = iindex ; i <= eindex ; i++ )
4502:     {
4503:        if ( cid == -1 ) return;
4504:
4505:        type = CANOBject [i][cid].type;
4506:        if ( type != REFINE )
4507:        {
4508:           sum += CANOBject [i][cid].grsum;
4509:           length += CANOBject [i][cid].length;
4510:        }
4511:        cid = CANOBject [i][cid].next id;
4512:     }
4513:
4514:     if (length == 0) return;
4515:     gravg = sum / length;
4516:
4517:     if ( gravg <= ( minedge + 8 ) )
4518:     {
4519:        ret = LeafConnection ( iindex, iid, eindex, eid );
4520:        if ( ret == -1 )
4521:           NearConnect ( iindex, iid, eindex, eid );
4522:     }
4523: }
4524:
4525: int  CRunLength::LeafConnection (int init, int iid, int end, int eid )
4526: {
4527:     int i, j, pid, nid;
4528:     int fedge, redge, pfedge, predge;
4529:     int nfedge, nredge;
4530:     int temp1, temp2;
4531:     CVt52App *PApp = (CVt52App *) AfxGetApp ();
4532:
4533:
4534:     fedge = CANOBject [init][iid].fedge;
4535:     redge = CANOBject [init][iid].redge;
4536:
4537:     i = init - 1;
4538:     for ( j = 0 ; j < OBCount [i] ; j++ )
4539:     {
4540:        if ( CANOBject [i][j].type != REFINE )
4541:        {
4542:           pfedge = CANOBject [i][j].fedge;
4543:           predge = CANOBject [i][j].redge;
4544:
4545:           if ( ( pfedge >= fedge && pfedge <= redge && predge >= redge ) ||
4546:                ( predge >= fedge && predge <= redge && pfedge <= fedge) )
4547:           {
4548:              pid = CANOBject [i][j].prev id;
4549:              nid = CANOBject [i][j].next id;
4550:              if ( pid == -1 || nid == -1 )
```

```
4551:                AddCrossPoint ( i, j, 1+1 );
4552:             else
4553:                CANOBject [i][j].cross = OBCROSS;
4554:
4555:             if ( vdebug )
4556:             {
4557:                temp1 = i * m_sample + m_xorg;
4558:                temp2 = (( pfedge + predge ) >> 1) + m_yorg;
4559:                if ( pid != -1 && nid != -1 )
4560:                   CIC draw cross ( oh Console [PApp->Frame Index],
4561:                      (double) temp1,   (double) temp2,
4562:                      5, 0, CDE PEN BLUE );
4563:                CIC draw line ( oh Console [PApp->Frame Index],
4564:                   (double) temp1, (double) temp2,  (double) (temp1+m_sample),
4565:                   (double) ( ((fedge+redge) >> 1 ) + m_yorg ),
4566:                   CDE PEN RED );
4567:             }
4568:             DrawLeaf ( init, iid, end, eid, CDE PEN BLUE );
4569:             return 1;
4570:          }
4571:       }
4572:    }
4573:
4574:    fedge = CANOBject [end][eid].fedge;
4575:    redge = CANOBject [end][eid].redge;
4576:    i = end + 1;
4577:    for ( j = 0 ; j < OBCount [i] ; j++ )
4578:    {
4579:       if ( CANOBject [i][j].type != REFINE )
4580:       {
4581:          nfedge = CANOBject [i][j].fedge;
4582:          nredge = CANOBject [i][j].redge;
4583:
4584:          if ( ( nfedge >= fedge && nfedge <= redge && nredge >= redge ) ||
4585:               ( nredge >= fedge && nredge <= redge && nfedge <= fedge) )
4586:          {
4587:             pid = CANOBject [i][j].prev_id;
4588:             nid = CANOBject [i][j].next_id;
4589:             if ( pid == -1 || nid == -1 )
4590:                AddCrossPoint ( i, j, i-1 );
4591:             else
4592:                CANOBject [i][j].cross = OBCROSS;
4593:
4594:             if ( vdebug )
4595:             {
4596:                temp1 = i * m_sample + m_xorg;
4597:                temp2 = (( nfedge + nredge ) >> 1) + m_yorg;
4598:                if ( pid != -1 && nid != -1 )
4599:                   CIC draw cross ( oh Console [PApp->Frame Index],
4600:                      (double) temp1,   (double) temp2,
4601:                      5, 0, CDE PEN BLUE );
4602:                CIC draw line ( oh Console [PApp->Frame Index],
4603:                   (double) temp1, (double) temp2,  (double) (temp1-m_sample),
4604:                   (double) ( ((fedge+redge) >> 1 ) + m_yorg ),
4605:                   CDE PEN RED );
4606:             }
4607:             DrawLeaf ( init, iid, end, eid, CDE PEN BLUE );
4608:             return 1;
4609:          }
4610:       }
4611:    }
4612:
4613:    return -1;
4614: }
4615:
4616: void CRunLength::NearConnect ( int init, int iid, int end, int eid )
4617: {
4618:    int i, j, id1, id2;
4619:    int temp1, temp2, fedge, redge;
4620:    int mpid, mcid, mnid;
4621:    int nid1, nid2;
4622:    int diff1, diff2, diff3;
4623:    int count, max, min, ipos, epos, minpos, minid;
4624:    CVt52App *PApp = (CVt52App *) AfxGetApp ();
4625:
```

```
4626:
4627:     max = 0;
4628:     ipos = -1;
4629:     for ( j = 0 ; j < OBCount [init] ; j++ )
4630:     {
4631:        count = CANOBject [init][j].count;
4632:        if ( count > max )
4633:        {
4634:           max = count;
4635:           ipos = j;
4636:        }
4637:     }
4638:
4639:     max = 0;
4640:     epos = -1;
4641:     for ( j = 0 ; j < OBCount [end] ; j++ )
4642:     {
4643:        count = CANOBject [end][j].count;
4644:        if ( count > max )
4645:        {
4646:           max = count;
4647:           epos = j;
4648:        }
4649:     }
4650:     // main branch id : id1, leaf id : id2
4651:     id1 = ipos;
4652:     id2 = iid;
4653:     mcid = NearId ( init, ipos, iid );
4654:     if ( mcid != ipos )
4655:     {
4656:        mpid = CANOBject [init][mcid].prev id;
4657:        if ( mpid == -1 ) mpid = CANOBject [init][mcid].prev_id;
4658:     }
4659:     else
4660:        mpid = CANOBject [init][mcid].prev id;
4661:     nid1 = CANOBject [init][ipos].next id;
4662:     nid2 = CANOBject [init][iid].next id;
4663:     if ( nid1 != -1 && nid2 != -1 )
4664:        mnid = NearId ( init+1, nid1, nid2 );
4665:
4666:     min = 10000;
4667:     for ( i = init ; i <= end ; i++ )
4668:     {
4669:        if ( id1 == -1 || id2 == -1 ) break;
4670:        fedge = CANOBject [i][id2].fedge;
4671:        redge = CANOBject [i][id2].redge;
4672:        if ( id1 > id2 )
4673:        {
4674:           diff1 = abs (CANOBject [i-1][mpid].fedge - redge);
4675:           diff2 = abs (CANOBject [i][mcid].fedge   - redge);
4676:           diff3 = abs (CANOBject [i+1][mnid].fedge - redge);
4677:        }
4678:        else
4679:        {
4680:           diff1 = abs (fedge - CANOBject [i-1][mpid].redge);
4681:           diff2 = abs (fedge - CANOBject [i][mcid].redge);
4682:           diff3 = abs (fedge - CANOBject [i+1][mnid].redge);
4683:        }
4684:
4685:        if ( min > diff1 )
4686:        {
4687:           min = diff1;
4688:           minpos = i-1;
4689:           minid  = mpid;
4690:        }
4691:        if ( min > diff2 )
4692:        {
4693:           min = diff2;
4694:           minpos = i;
4695:           minid  = mcid;
4696:        }
4697:        if ( min > diff3 )
4698:        {
4699:           min = diff3;
4700:           minpos = i+1;
```

```
4701:            minid  = mnid;
4702:         }
4703:
4704:         id1 = nid1;
4705:         id2 = nid2;
4706:         mpid = mcid;
4707:         mcid = mnid;
4708:
4709:         if ( i == ( end - 1 ) )
4710:            mnid = CANOBject [i+1][id1].next id;
4711:         else if ( i < ( end - 1 ) )
4712:         {
4713:            nid1 = CANOBject [i+1][id1].next id;
4714:            nid2 = CANOBject [i+1][id2].next id;
4715:            if ( nid1 != -1 && nid2 != -1 )
4716:               mnid = NearId ( i+1, nid1, nid2 );
4717:         }
4718:      }
4719:
4720:      if ( min > 15 ) return;
4721:      mpid = CANOBject [minpos][minid].prev id;
4722:      mnid = CANOBject [minpos][minid].next id;
4723:      if ( mpid == -1 && mnid != -1 )
4724:         AddCrossPoint ( minpos, minid, minpos-1 );
4725:      else if ( mpid != -1 && mnid == -1 )
4726:         AddCrossPoint ( minpos, minid, minpos+1 );
4727:      else if ( mpid != -1 && mnid != -1 )
4728:      {
4729:         CANOBject [minpos][minid].cross = OBCROSS;
4730:
4731:         if ( vdebug )
4732:         {
4733:            int fedge, redge;
4734:            fedge = CANOBject [minpos][minid].fedge;
4735:            redge = CANOBject [minpos][minid].redge;
4736:            temp1 = minpos * m sample + m xorg;
4737:            temp2 = (( fedge + redge ) >> 1) + m yorg;
4738:            CIC draw cross ( oh Console [PApp->Frame Index],
4739:               (double) temp1,   (double) temp2,
4740:               5, 0, CDE PEN BLUE );
4741:            if ( minpos == ( init -1 ) || minpos == ( end + 1 ) )
4742:            {
4743:               if ( minpos == (init - 1) )
4744:               {
4745:                  fedge = CANOBject [init][iid].fedge;
4746:                  redge = CANOBject [init][iid].redge;
4747:                  i = temp1 + m sample;
4748:               }
4749:               else
4750:               {
4751:                  fedge = CANOBject [end][eid].fedge;
4752:                  redge = CANOBject [end][eid].redge;
4753:                  i = temp1 - m sample;
4754:               }
4755:
4756:               CIC draw line ( oh Console [PApp->Frame Index],
4757:                  (double) temp1, (double) temp2,  (double) i,
4758:                  (double) ( ((fedge+redge) >> 1 ) + m_yorg ),
4759:                  CDE PEN RED );
4760:            }
4761:         }
4762:      }
4763:      DrawLeaf ( init, iid, end, eid, CDE PEN BLUE );
4764: }
4765:
4766: int  CRunLength::NearId ( int index, int mid, int lid )
4767: {
4768:    int i, idiff, pid, cid, nid;
4769:
4770:    idiff = abs ( mid - lid );
4771:    if ( idiff > 1 )
4772:    {
4773:       for ( i = 1 ; i < idiff ; i++ )
4774:       {
4775:          if ( mid > lid ) cid = lid + i;
```

```
4776:            else          cid = lid - i;
4777:            pid = CANObject [index][cid].prev id;
4778:            nid = CANObject [index][cid].next id;
4779:            if ( pid != -1 || nid != -1 )
4780:               return cid;
4781:         }
4782:      }
4783:
4784:      return mid;
4785: }
4786:
4787: void CRunLength::AddCrossPoint ( int nindex, int nid, int cindex )
4788: {
4789:    int next id, count, ncount;
4790:    int i, prev id, pcount;
4791:    CVt52App *PApp = (CVt52App *) AfxGetApp ();
4792:
4793:    if ( nindex > cindex )
4794:    {
4795:       for ( i = 0 ; i < 10 ; i++, nindex++)
4796:       {
4797:          count = CANObject [nindex][nid].count;
4798:          next id = CANObject [nindex][nid].next id;
4799:
4800:          if ( next_id == -1 ) break;
4801:
4802:          ncount = CANObject [nindex+1][next id].count;
4803:          if ( ncount > (count+1) )
4804:          {
4805:             CANObject [nindex+1][next id].cross = OBCROSS;
4806:             if (vdebug)
4807:                CIC draw cross ( oh Console [PApp->Frame Index],
4808:                   (double) ((nindex+1)*m sample+m xorg),
4809:                   (double) (((CANObject[nindex+1][next id].fedge +
4810:                            CANObject[nindex+1][next id].redge) >>1) + m yorg
     ),
4811:                   5, 0, CDE PEN BLUE );
4812:             break;
4813:          }
4814:
4815:       }
4816:    }
4817:    else
4818:    {
4819:       for ( i = 0 ; i < 10 ; i++, nindex--)
4820:       {
4821:          count = CANObject [nindex][nid].count;
4822:          prev id = CANObject [nindex][nid].prev id;
4823:
4824:          if ( prev_id == -1 ) break;
4825:
4826:          pcount = CANObject [nindex-1][prev id].count;
4827:          if ( pcount > (count-1) )
4828:          {
4829:             CANObject [nindex-1][prev id].cross = OBCROSS;
4830:             if (vdebug)
4831:                CIC draw cross ( oh Console [PApp->Frame Index],
4832:                   (double) ((nindex-1)*m sample+m xorg),
4833:                   (double) (((CANObject[nindex-1][prev id].fedge +
4834:                            CANObject[nindex-1][prev id].redge) >>1) + m yorg
     ),
4835:                   5, 0, CDE PEN BLUE );
4836:             break;
4837:          }
4838:       }
4839:
4840:    }
4841: }
4842: void CRunLength::FindCuttingPoint ()
4843: {
4844:    int i, j, xc, cross;
4845:    CVt52App *PApp = (CVt52App *) AfxGetApp ();
4846:
4847:    PApp->mcutcnt = 0;
4848:    for ( i = 0, xc = m_xorg ; i < xcount ; i++, xc += m_sample )
```

```
4849:     {
4850:        for ( j = 0 ; j < OBCount [i] ; j++ )
4851:        {
4852:           cross = CANObject [i][j].cross;
4853:           if ( cross == OBCROSS || cross == OBPARENT )
4854:              CALCuttingPoint ( i , j );
4855:        }
4856:     }
4857:
4858:     CIC force refresh ( oh Console[PApp->Frame Index] );
4859: }
4860:
4861: void CRunLength::CALCuttingPoint ( int i, int j )
4862: {
4863:     int index, conti, ret;
4864:     int cid, cross, init, color;
4865:     int CHAIN [100], temp;
4866:     int xcut, ycut, next_id;
4867:     double xpnt, ypnt;
4868:     CVt52App *PApp = (CVt52App *) AfxGetApp ();
4869:
4870:     index = i;
4871:     cid = j;
4872:
4873:     conti = 1;
4874:     for ( int k = 0 ; index < xcount ; index++, k++ )
4875:     {
4876:        cross = CANObject [index][cid].cross;
4877:        if ( conti )
4878:        {
4879:           if ( cross != OBCROSS && cross != OBPARENT )
4880:           {
4881:              init = k;
4882:              conti  = 0;
4883:           }
4884:           else
4885:              if ( k ) CANObject [index][cid].cross += OBCHECK;
4886:
4887:        }
4888:        else
4889:           // Cutting Point
4890:           if ( cross == OBCROSS || cross == OBPARENT )
4891:           {
4892:              int avg;
4893:              avg = init + (( k - init ) >> 1);
4894:              xcut = i + avg;
4895:              temp = CHAIN [avg];
4896:              if (temp == -1)
4897:              {
4898:                 xcut++;
4899:                 temp = CHAIN[avg+1];
4900:              }
4901:              ycut = CANObject [xcut][temp].fedge +
4902:                    CANObject [xcut][temp].redge;
4903:              ycut >>= 1;
4904:              if ( CANObject [xcut][temp].type == NORMAL )
4905:              {
4906:                 xpnt = (double) (xcut*m_sample + m_xorg);
4907:                 ypnt = (double) (ycut + m_yorg);
4908:                 PApp->xCutPnt[PApp->mcutcnt] =
4909:                    (int) (( PApp->mcal_xcent - xpnt ) * PApp->m_xscale);
4910:                 PApp->yCutPnt[PApp->mcutcnt] =
4911:                    (int) (( ypnt - PApp->mcal_ycent ) * PApp->m_yscale);
4912:                 PApp->mcutcnt++;
4913:                 if ( CANObject [xcut][temp].avg >= (maxedge - 10) )
4914:                    color = CDE_PEN_YELLOW;
4915:                 else color = CDE_PEN_ORANGE;
4916:                 if (vdebug)
4917:                 {
4918:                    CIC draw_line ( oh Console[PApp->Frame Index],
4919:                       (double) (xcut*m_sample+m_xorg-1), (double)
      (ycut+m_yorg-10),
4920:                       (double) (xcut*m_sample+m_xorg-1),
4921:                       (double) (ycut+m_yorg+10), color );
4922:                    CIC draw_line ( oh Console[PApp->Frame Index],
```

```
4923:                         (double) (xcut*m_sample+m_xorg), (double)
      (ycut+m yorg-10),
4924:                         (double) (xcut*m sample+m xorg),
4925:                         (double) (ycut+m yorg+10), color );
4926:                     CIC draw line ( oh Console[PApp->Frame Index],
4927:                         (double) (xcut*m_sample+m_xorg+1), (double)
      (ycut+m yorg-10),
4928:                         (double) (xcut*m sample+m xorg+1),
4929:                         (double) (ycut+m_yorg+10), color );
4930:                 }
4931:              }
4932:              //if ( cross == OBPARENT )
4933:                 ClearOBPARENT ( index, cid );
4934:              break;
4935:           }
4936:
4937:        next id = CANOBject [index][cid].next id;
4938:        if ( next_id == -1 )
4939:        {
4940:           ret = CheckStop ( index, cid );
4941:           if ( ret == -1 ) break;
4942:           else
4943:           {
4944:              CHAIN[k++] = cid;
4945:              CHAIN[k] = -1;
4946:              index++;
4947:              cid = ret;
4948:           }
4949:        }
4950:        else
4951:        {
4952:           CHAIN[k] = cid;
4953:           cid = next id;
4954:        }
4955:     }
4956: }
4957:
4958: void CRunLength::ClearOBPARENT ( int i, int j )
4959: {
4960:     int index, cid;
4961:     int next_id, cross;
4962:     int pid;
4963:
4964:     index = i;
4965:     cid   = j;
4966:
4967:     pid = -2;
4968:     for ( ; index < xcount ; index++ )
4969:     {
4970:        cross = CANOBject [index][cid].cross;
4971:
4972:        if ( cross == OBPARENT )
4973:        {
4974:           if ( pid != -2 )
4975:              CANOBject [index-1][pid].cross += OBCHECK;
4976:           pid = cid;
4977:        }
4978:        else
4979:           break;
4980:
4981:        next id = CANOBject [index][cid].next id;
4982:
4983:        if ( next_id == -1 ) break;
4984:
4985:        cid = next id;
4986:     }
4987: }
4988:
4989: int CRunLength::CheckStop ( int index, int cid )
4990: {
4991:     int pcent, plength;
4992:     int ncent, nlength;
4993:     int i, nid, diff, min;
4994:
4995:     if ( index > ( xcount - 4 ) ) return -1;
```

```
4996:     if ( OBCount [index+1] ) return -1;
4997:     if ( OBCount [index+1] == 0 )
4998:     {
4999:        pcent = CANOBject[index][cid].fedge +
5000:                CANOBject[index][cid].redge;
5001:        plength = CANOBject[index][cid].length;
5002:
5003:        nid = -1; min = 1000;
5004:        for ( i = 0 ; i < OBCount[index+2] ; i++ )
5005:        {
5006:           ncent = CANOBject[index+2][i].fedge +
5007:                   CANOBject[index+2][i].redge;
5008:           nlength = CANOBject[index+2][i].length;
5009:           if ( abs (pcent-ncent) < 6 && abs (plength-nlength) < 3 )
5010:           {
5011:              diff = abs (pcent-ncent) + abs (plength-nlength);
5012:              if ( min > diff )
5013:              {
5014:                 nid = i;
5015:                 min = diff;
5016:              }
5017:           }
5018:        }
5019:     }
5020:
5021:     if ( nid != -1 ) return nid;
5022:     else             return -1;
5023: }
5024:
5025: BOOL CRunLength::OnInitDialog()
5026: {
5027:     CVt52App *PApp = (CVt52App *) AfxGetApp ();
5028:
5029:     CDialog::OnInitDialog();
5030:
5031:     // TODO: Add extra initialization here
5032:     m xorg   = PApp->mRunxorg;
5033:     m xsize  = PApp->mRunxsize;
5034:     m yorg   = PApp->mRunyorg;
5035:     m ysize  = PApp->mRunysize;
5036:     m sample = PApp->mRunsample;
5037:     m diff   = PApp->mRundiff;
5038:     thres = 90;
5039:
5040:     UpdateData(FALSE);
5041:
5042:     if ( vdebug )
5043:     {
5044:        CIC draw rect ( oh Console[PApp->Frame Index], (double) m xorg,
5045:                  (double) m yorg, (double) m_xsize, (double) m_ysize,
5046:                  CDE PEN GREEN );
5047:        CIC force refresh ( oh Console[PApp->Frame Index] );
5048:     }
5049:
5050:     return TRUE;  // return TRUE unless you set the focus to a control
5051:                   // EXCEPTION: OCX Property Pages should return FALSE
5052: }
```